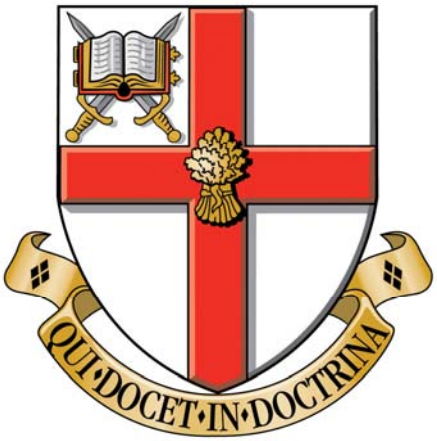

University of Chester

The Numerical Solution of Fractional and Distributed Order Differential Equations

Thesis submitted in accordance with the requirements of the University of Liverpool for the degree of Doctor in Philosophy by Joseph Arthur Connolly

December 2004

Declaration

No part of the work referred to in this thesis has been submitted in support of an application for another degree or qualification of this or any other institution of learning.

Acknowledgements

I acknowledge the help and guidance of my supervisors Professor Neville J. Ford and Doctor John T. Edwards.

I thank Chester College for its financial support in the form of a Chester College research student bursary.

Abstract

Fractional Calculus can be thought of as a generalisation of conventional calculus in the sense that it extends the concept of a derivative (integral) to include non-integer orders. Effective mathematical modelling using Fractional Differential Equations (FDEs) requires the development of reliable flexible numerical methods.

The thesis begins by reviewing a selection of numerical methods for the solution of Single-term and Multi-term FDEs.

We then present:

1. a graphical technique for comparing the efficiency of numerical methods. We use this to compare Single-term and Multi-term methods and give recommendations for which method is best for any given FDE.

2. a new method for the solution of a non-linear Multi-term Fractional Differential Equation.

3. a sequence of methods for the numerical solution of a Distributed Order Differential Equation.

4. a discussion of the problems associated with producing a computer program for obtaining the optimum numerical method for any given FDE.

Contents

Chapter 1

Introduction

1.1 Aim of Thesis:

To enhance an engineer's ability to use fractional calculus as a modelling tool.

1.2 Motivation

In recent years fractional calculus has emerged as an important area in modelling many physical phenomena. A restriction to the wide uptake of fractional and distributed order differential equations in modelling is the lack of efficient reliable methods for their solution. Where methods are available, it is often unclear as to which method is best for any given equation.

1.3 Objectives

(1) To compare the numerical efficiency of single-term, multi-term and distributed fractional differential equation (FDE) methods.

(2) To produce new methods, that improve on the numerical efficiency of current methods.

(3) To produce recommendations for mathematicians/scientists on which numerical method is most economical for a given FDE.

(4) To discuss the problems associated with creating an automated computer program which finds the optimum method, and implements that method, for a given equation.

1.4 Results

A new graphical technique for determining a numerical method's efficiency is presented. This technique is used to compare numerical methods for single-term, multi-term and distributed order problems. We give recommendations for scientists/mathematicians on which method is most numerically efficient for any given FDE, but show that an automated system may be more desirable. The optimum number of corrector iterations for single and multi-term predictor corrector based methods are determined experimentally.

We introduce a numerically efficient algorithm for the solution of multi-term FDEs. The validity of this algorithm is proved.

A new Richardson extrapolation procedure, for accelerating the convergence order of the distribution integral of a distributed order differential equation (DODE), is presented.

A sequence of distributed extended trapezoidal rules are presented. These produce dramatic savings in computational run-time for DODEs.

A computer program is produced, which automatically computes and implements the most numerically efficient method for any single-term, multi-term or distributed order differential equation.

Chapter 2

We examine the question, what is a fractional derivative? A brief history of fractional calculus is presented. We exhibit the different definitions that fractional derivatives and integrals can have. We introduce the concept of a fractional differential equation and show the various forms with which we will be concerned.

Chapter 3

A brief survey of the possible uses of fractional differential equations is presented. Specific examples from recent applications enable us to understand the variety of fields in which fractional calculus can be of use.

Chapter 4

Sufficient relevant theory is given to allow the thesis to be self contained. We give analytical solutions to the various fractional differential equations with which we are concerned and explain why numerical approximations are necessary.

Chapter 5

We describe 5 different algorithms for the numerical solution of single-term FDEs. All 5 are suitable for linear equations, only one is specifically designed for non-linear problems.

Chapter 6

We introduce a graphical technique for determining single-term FDE methods' relative numerical efficiency. We show that the predictor corrector method's optimum number of corrector iterations depends upon the order of the fractional derivative. A comparison of the five methods described in chapter 5 is performed. We show that the most efficient numerical method depends upon the order of the fractional derivative and the content of any functions involved.

Chapter 7

An exposition of current algorithms available for the numerical solution of multi-term FDEs is given. A 'new' algorithm specifically designed to give good numerical performance for non-linear equations, but which is also numerically efficient for some classes of linear problems, is introduced. We prove the convergence of the 'new' method.

Chapter 8

The optimum number of corrector iterations for both the 'new' method and the predictor corrector method are calculated for the various types of equation and orders of derivative. We show how the stability of the multi-term predictor corrector and the 'new' method can be affected by constants above a certain threshold. A comparison of the various multi-term methods' numerical efficiency is performed.

Chapter 9

We introduce the concept of a distributed order differential equation (DODE). We give an exposition of the Diethelm and Ford [14] scheme for the numerical solution of DODEs. We show various versions of Diethelm and Ford's scheme and compare the relative merits of each.

Chapter 10

A new DODE Richardson extrapolation is introduced. The existence of an asymptotic error expansion (AEE) is shown. The increase in performance is analysed.

Chapter 11

Extended trapezoidal rules (ETRs) are introduced. We show how a fourth order ETR can be used, together with a trapezoidal starting and ending procedure, to discretize the order distribution of a DODE. The convergence order of this method is shown to be $O(h^3)$. We show how fourth and sixth order ETRs can be applied directly to a subset of DODEs.

Chapter 12

We give the fractional differential equation solver's design schematics. We show how the program has been produced in a modular format, so that additional methods or more efficient code can be substituted seamlessly. We discuss various programming issues and give example code where necessary.

Chapter 13

In the final chapter we discuss current and future work. Variable order differential equations are introduced, possible applications are discussed and a prototype algorithm for their solution is given. Partial FDEs and partial DODEs are introduced. The possibility of a tandem computer program for the determination of parameters used in the modelling of FDEs is discussed.

Chapter 2

What are Fractional Differential Equations?

History of the Fractional Calculus

The concept of differentiation and integration of non-integer order dates from the origin of calculus itself. Correspondence by Leibniz in the mid 17th century considered the idea of a derivative of order 1/2. It was Abel however, who in papers with Liouville in 1823 used the fractional calculus to solve the tautochrome problem, and thus made the first notable contribution.

The 'real creator of fractional calculus' [60] is widely regarded to be Liouville, who in 1832-1837 produced a series of papers which laid down the foundations for fractional integro-differentiation. In 1847 Riemann wrote a paper, only published in 1876, 10 years after his death. In this paper Riemann defined the fractional integral, and it is this definition combined with that of Liouville which has become the most widely used.

Grünwald and Letnikov (1867-68) used a finite difference approach for defining the fractional calculus. This definition was proved to be equivalent to the Riemann-Liouville derivative when the fractional difference was interpreted appropriately.

In 1967 Caputo [5] defined a version of the fractional derivative similar to Riemann-Liouville type, in which he changed the order of integration and differentiation enabling the user to apply non-zero initial conditions to the fractional integral with real life measurable quantities. In recent years this definition has become popular in the area of algorithm development.

Excellent historical summaries can be found in Miller and Ross [54] and Samko, Kilbas and Marichev [60]. In [56], Podlubny gives a very readable summary of

basic techniques and in [58] gives a geometric and physical interpretation of the fractional derivative. In a series of papers [43], [44], [46], [47], [48] and [49], Lorenzo and Hartley discuss the various issues and possible future directions of the fractional calculus.

Why is Fractional Calculus Important?

In ordinary differential equations, if the values of a problem's derivatives are known at a particular instance, the value of the solution at that instance can be determined. In fractional differential equations previous values of the solution and the derivatives are required to obtain a solution at a particular instance. The memory effect of the convolution in the fractional integral gives the equation increased expressive power. This allows the modelling of a variety of physical behaviour, such as viscoelasticity [32], diffusion [34], viscoplasticity, as well as more abstract concepts such as mappings using tensor fields [22].

If multiple instances of derivatives of different orders are added, we obtain composite behaviour. These multi-term and distributed order equations have recently been the subject of models from a variety of industrial fields [6], [7] and [35]. In chapter 3 we give a brief synopsis on the practical present and future uses of fractional calculus.

Why is Fractional Calculus Numerically Challenging?

As stated earlier fractional derivatives possess a memory effect. This can be very useful in modelling behaviour, but can also be computationally expensive.

Any numerical approximation of a fractional differential equation requires an exponential increase in work with increasing time. If multiple instances of fractional derivatives are included the workload again increases exponentially.

Small differences between various algorithms' performances over short run times can change to large differences if the run time is increased. Different algorithms may be more efficient depending upon the accuracy of the result required. For some algorithms the structure of the fractional differential equation can drastically change the computational cost.

If the wide uptake of fractional calculus into the scientific community is to take place solutions must be obtained while still of use. A better understanding of algorithm performance would allow the user to implement the most efficient algorithm for a given equation.

2.1 Fractional Derivatives

We shall first introduce the gamma function used in several definitions. When not explicitly stated all definitions are taken from Samko, Kilbas and Marichev [60]. If definitions and theorems are taken from books, page numbers are given after the bibliography number.

2.1.1 The Gamma Function

The gamma function is a generalization of the factorial function $n!$. Where the factorial is only applicable to positive integer order n, the gamma function can be used for any real number.

Definition 2.1.1 *([60] 15)*

$$\Gamma(z) = \int_0^\infty t^{z-1}e^{-t}dt, Re\ z > 0 \tag{2.1}$$

$$\Gamma(z+1) = z\Gamma(z) \tag{2.2}$$

Sometimes it is convenient to use the factorial notation $\alpha! = \Gamma(\alpha + 1)$ where $\alpha \notin \mathbb{N}$. We can then write the binomial coefficient as

$$\binom{-z}{\zeta} = \frac{\Gamma(1-z)}{\Gamma(\zeta+1)\Gamma(1-z-\zeta)}.$$

2.2 The Riemann-Liouville Fractional Calculus

The Riemann-Liouville definition of fractional calculus has historical precedent [60] over the other definitions.

Since $(n-1)! = \Gamma(n)$, the well known formula for an n-fold integral

$$\int_a^t dt \int_a^t dt \ldots \int_a^t \varphi(t)dt = \frac{1}{(n-1)!}\int_a^t (t-u)^{n-1}\varphi(u)du$$

leads naturally to the definition of the left and right sided Riemann-Liouville fractional integrals.

Definition 2.2.1 *([60] 33)*

$$(J_{a+}^{\alpha}\varphi)(t) =^{def} \frac{1}{\Gamma(\alpha)}\int_a^t \frac{\varphi(u)}{(t-u)^{1-\alpha}}du, t > a, \tag{2.3}$$

$$(J_{b-}^{\alpha}\varphi)(t) =^{def} \frac{1}{\Gamma(\alpha)}\int_t^b \frac{\varphi(u)}{(u-t)^{1-\alpha}}du, t < a \tag{2.4}$$

where $\alpha > 0$.

The Riemann-Liouville Fractional Derivative

The Riemann-Liouville fractional derivative of order $\alpha > 0$, takes the form

Definition 2.2.2 *([60] 35)*

$$(D_*^{\alpha}y)(t) = \frac{1}{\Gamma(m-\alpha)}\frac{d^m}{dt^m}\int_0^t \frac{y(u)}{(t-u)^{\alpha-m+1}}du$$

where m is the integer defined by $m-1 < \alpha \leq m$.

The notation can be used,

$$D_{a+}^{\alpha}y = J_{a+}^{-\alpha}y$$

assuming each means the derivative.

2.2.1 Semi-Group property

Theorem 2.2.3 *([60] 46)*

The relation

$$J_{a+}^{\alpha}J_{a+}^{\beta}\varphi = J_{a+}^{\alpha+\beta}\varphi$$

is valid in any of the following cases:

1. *Re $\beta > 0$, Re $(\alpha+\beta) > 0, \varphi(x) \in L_1(a,b)$;*
2. *Re $\beta < 0$, Re $\alpha > 0, \varphi(x) \in I_{a+}^{-\beta}(L_1)$;*
3. *Re $\alpha < 0$, Re$(\alpha+\beta) < 0, \varphi(x) \in I_{a+}^{-\alpha-\beta}(L_1)$,*

the cases $\alpha = 0, \beta = 0$ and $\alpha+\beta = 0$ being also admissible for real α and β.

2.3 The Grünwald and Letnikov Definition

If a function $f(t)$ is differentiable up to order n, it is well known that the finite difference can be defined as

$$f^{(n)}(t) = \lim_{h \to 0} \frac{(\nabla_h^n f)(t)}{h^n}. \tag{2.5}$$

The approach is a generalization of the well known backward difference operator for ordinary differential equations (ODEs). By substituting the fractional derivative α for the integer order derivative, the approximation of the α-th derivative is

Definition 2.3.1 *([60] 371)*

$$h^{-\alpha} \nabla_h^\alpha y(t) = \tilde{D}^\alpha y(t) = h^{-\alpha} \sum_{j=0}^{[t/h]} (-1)^j \binom{\alpha}{j} y(t - jh).$$

2.4 The Caputo Definition

The Riemann-Liouville derivative has the drawback that initial conditions of fractional differential equations must be defined in the form

$$\frac{d^{\alpha-k}}{dt^{\alpha-k}} y(t)|_{t=0+} = b_k, \quad k = 1, 2, \ldots, m-1 = \lfloor \alpha \rfloor, \tag{2.6}$$

with given values b_k. In practical applications these are generally not available and often no physical meaning exists.

Definition 2.4.1 *([56])*

The Caputo derivative is defined as

$$D^\alpha f(t) = \frac{1}{\Gamma(n-\alpha)} \int_a^t \frac{f^{(n)}(\tau) d\tau}{(t-\tau)^{\alpha+1-n}}, (n-1 < \alpha < n)$$

where α may be an arbitrary positive real number.

Blank [2] states, if $\alpha \geq 0$, $u > -1$ and $t > 0$, we have

$$D^\alpha t^u = \frac{t^{(n-\alpha)} \Gamma(u+1)}{\Gamma(u+1-\alpha)}. \tag{2.7}$$

This definition replaces the need for initial conditions of the form (2.6), with those of classical form,

$$y^{(k)}(0) = y_0^{(k)}, \quad k = 0, 1, \ldots, m-1.$$

This convenience has lead to the Caputo fractional derivative becoming the popular choice for mathematicians and scientists.

The Riemann-Liouville and Caputo definitions are linked by,

$$D^{\alpha} z(t) = J_{a+}^{m-\alpha} D_*^m z(t)$$

where $m := \lceil \alpha \rceil$ is the smallest integer $\geq \alpha$.

2.5 Fractional Differential Equations

2.5.1 Single-term FDEs [19]

The single-term non-linear FDE is defined as:

$$D^{\alpha} y(t) = f(t, y(t)), \tag{2.8}$$

where $0 < \alpha < m, m = \lceil \alpha \rceil$, subject to the initial conditions

$$y^{(k)}(0) = y_0^{(k)}, k = 0, \ldots, m-1. \tag{2.9}$$

The linear version is defined as:

$$D^{\alpha} y(t) = \lambda y(t) + g(t), \tag{2.10}$$

where $0 < \alpha < m, m = \lceil \alpha \rceil$, subject to the initial conditions (2.9).

2.5.2 Multi-term FDEs [28]

The non-linear multi-term equation is defined as:

$$D^{\alpha_N} y(t) = f(t, y(t), D^{\alpha_1} y(t), \ldots, D^{\alpha_{N-1}} y(t)) \tag{2.11}$$

equipped with initial conditions,

$$y^{(k)}(0) = y_0^{(k)}, k = 0, 1, \ldots, \lceil \alpha_N \rceil - 1. \tag{2.12}$$

The linear multi-term FDE is defined as:

$$\left[D^{\alpha_N} + b_{N-1} D^{\alpha_{N-1}} + \ldots + b_1 D^{\alpha_1} + b_0 D^0 \right] y(t) = g(t), \tag{2.13}$$

with initial conditions (2.12).

2.5.3 Distributed Order Differential Equations [6]

Caputo [6] defines a general linear distributed order differential equation as:

$$\sum_{n=1}^{m} \lim_{\epsilon_n \to 0} \int_{\epsilon_n}^{1} b_n(\alpha) I^{\alpha} u^{(n)}(t) d\alpha + \sum_{n=0}^{m} c_n u^{(n)}(t) = g(t). \tag{2.14}$$

2.6 Conclusions

In this chapter we have given a brief history of fractional calculus, explained why fractional calculus is important and why it is numerically challenging. We have stated the different definitions of fractional derivative and introduced the concept of fractional and distributed order differential equations.

In the next chapter we show some practical applications of FDEs and DODEs.

Chapter 3

Applications of Fractional Calculus

This chapter focuses on the uses of fractional differential equations (FDEs) in modelling physical phenomena. We first discuss several uses of linear and non-linear single and multi-term FDEs. The concept of distributed order differential equations (DODEs) is introduced.

3.1 Viscoelasticity

We follow the exposition given by Podlubny in [56].

Hooke's law states, for ideal solids, stress ρ is proportional to strain ϵ, thus

$$\rho(t) = E\varepsilon(t), \tag{3.1}$$

and for ideal Newtonian fluids, stress is proportional to the rate of change of extension, thus

$$\rho(t) = \eta \frac{d\varepsilon(t)}{dt}, \tag{3.2}$$

where E and η are constants.

Podubny notes that the stress is proportional to the zeroth derivative of strain for solids and the first derivative of strain for fluids, and shows that the intermediate stress may be proportional to an intermediate stress derivative of non-integer order,

$$\rho(t) = ED^{\alpha}\varepsilon(t), \tag{3.3}$$

where E and α are constants.

3.2 Viscoelastic Damped Vibration

In the aerospace industry, to stop unwanted motion in gas turbine fan blades, viscoelastic damping is applied. Hartley and Lorenzo [43] illustrate how fractional calculus can be used to model the damping required.

Figure 3.1 shows a spring-mass-visco damped dynamic system. The required function requires zero initializations, thus the equations describing figure 3.1 are given by:

$$F_1 = k_1 D^{q_1}(x_0 - x_{m1})$$

$$F_1 = k(x_{m1} - x_i)$$

$$F_2 = -k_2 D^{q_2}(x_0 - x_{m2})$$

$$F_2 = k_3 D^{q_3}(x_{m2} - x_i)$$

$$F_1 + F_2 = mD^2 x_0$$

Where

m = mass

F = force

x = displacements

k = damping coefficients

The q values can vary between 0 and 1 representing a continuum from a spring of infinite stiffness to a dashpot. For any given mass and force, the displacement can be controlled by manipulating the q values.

The above is a vector single-term FDE.

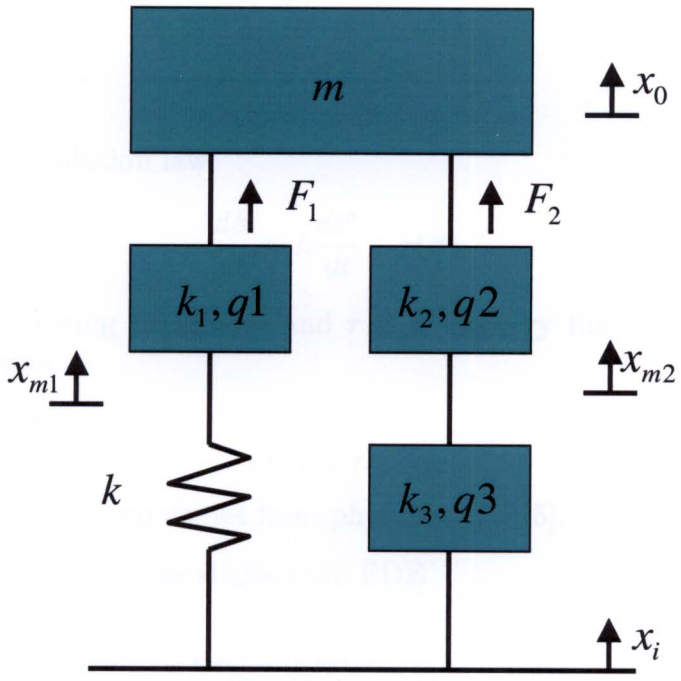

Figure 3.1: Viscoelastic damped vibration

3.3 Viscoplasticity

Viscoplasticity (A. Freed, Informal communication, 2004) is used as a model for rate-dependent plasticity. This is important for (high-speed) transient plasticity calculations. Viscoplasticity influences the stresses via the plastic strains

$$\sigma = E(\epsilon - \epsilon^p)$$

$$\frac{d\epsilon^p}{dt} = [f(\sigma - \beta)](\sigma - \beta)$$

σ stress

ϵ strain

ϵ^p plastic strain

E modulus

β back stress (internal or hidden variable)

Example of f:

$$f = a|\sigma - \beta|^n$$

where a, n are constants.

β is governed by evolution law,

$$\frac{d\beta}{dt} = h\frac{d\epsilon^p}{dt} - r(\beta)$$

where h is a hardening parameter and r is a recovery function which can be thermal or dynamic.

For thermal recovery

$$r(\beta) = r_0\beta^4.$$

The power 4 of the β term comes from physics $\in [3-5]$.

This results in the non-linear single-term FDE

$$D_*^\alpha \beta = hD_*^\alpha \epsilon^p - r(\beta).$$

3.4 Bagley-Torvik Equation

The motion of a rigid plate, mass (M) and area (S), connected by a massless spring of stiffness (K), immersed in a Newtonian fluid, was originally proposed by Bagley and Torvik [1].

In this section we follow the exposition of Podlubny [56].

We let ρ be the fluid density, μ be the viscosity and $v(t,z)$ be the transverse velocity, which is a function of time t and the distance from the plate, z. The displacement of the plate, y, is described by,

$$MD^2y(t) = g(t) - Ky(t) - 2S\sigma(t,0).$$

where

$$\sigma(t,z) = \sqrt{\mu\rho}D^{1/2}v(s,z).$$

Therefore,

$$AD^2y(t) + BD^{3/2}y(t) + Cy(t) = g(t), \quad y(0) = y'(0) = 0,$$

where, $A = M$, $B = 2S\sqrt{\mu\rho}$, and $C = K$.

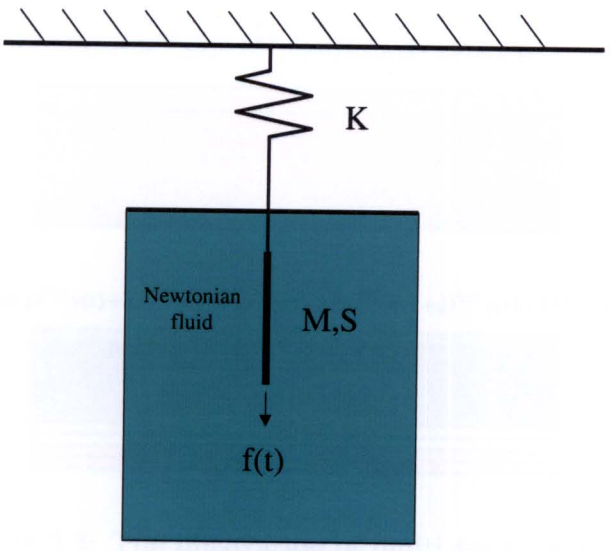

Figure 3.2: An immersed plate in a Newtonian fluid

3.5 Distributed Order Differential Equations

Diethelm and Ford [14] states that a distributed order differential equation (DODE) has the form,

$$\sum_{n=1}^{N} \lim_{\epsilon_n \to 0} \int_{\epsilon_n}^{1} b_n(\alpha) I^{\alpha} u^{(n)}(t) d\alpha + \sum_{n=0}^{N} c_n u^{(n)}(t) = g(t). \tag{3.4}$$

The concept of a distributed order differential equation (DODE) can be thought of as generalizing a multi-term equation. We consider a tile made up of layers of different viscoelastic components, each layer can be modelled individually by a fractional derivative, and any thickness or density issues by appropriate constants. In conjunction the tile can be modelled by a multi-term fractional differential equation comprising of the individual fractional derivatives. Consider a tile with the particular property that the fractional derivatives of sequential layers possess orders which are monotonic. As we introduce progressively more layers, illustrated in figure 3.3, in the limiting case, as the thickness of layers approaches 0, a DODE is generated.

In practice the engineering of such a tile is unlikely. The same material properties

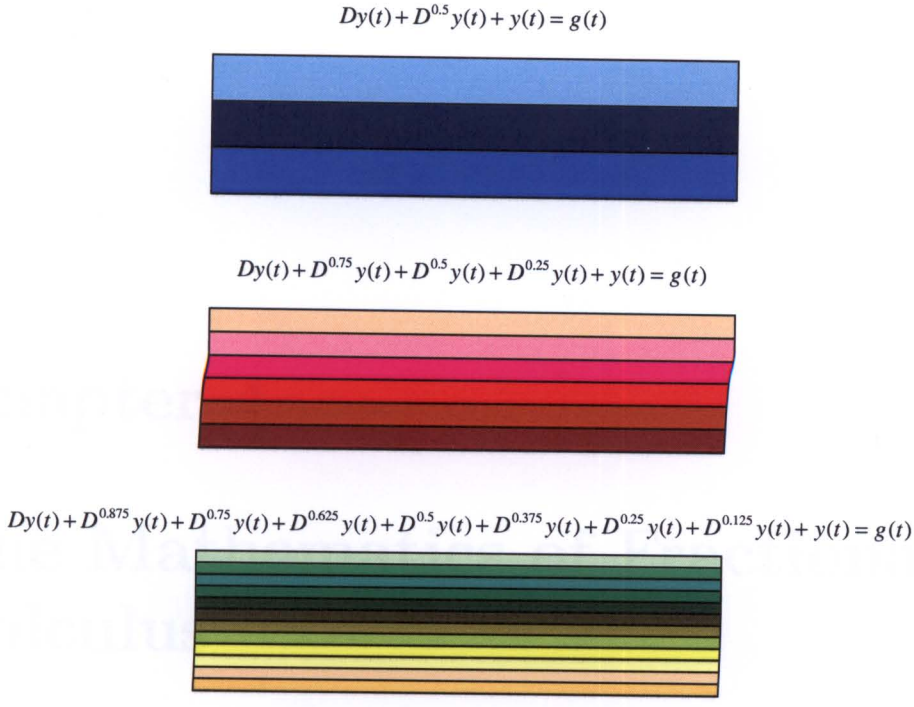

Figure 3.3: Tile illustration of multi-term problem

can be produced however, by applying a temperature gradient to a tile made up of thermo-viscoelastic material.

Chechkin, Gorenflo and Sokolov [7] and [8] show how DODEs can also be used to model super-diffusion and diffusion.

3.6 Conclusions

In this chapter we have given a brief insight into the many applications of fractional differential equations. Additional examples can be found in Podlubny [56], Debnath [10], Kleinz and Osler[39], and Schmidt and Gaul [61]

The concept of a distributed order differential equation has been introduced.

In the next chapter we state enough mathematics concerning the fractional calculus so as to make this thesis self contained.

Chapter 4

The Mathematics of Fractional Calculus

In this chapter we present sufficient relevant mathematics, so that this thesis is self contained. We show how the analytical solutions of fractional differential equations (FDEs) can be represented using various functions. We exhibit the various mathematical techniques that have been used to produce the algorithms which we use. The existence and uniqueness of FDE solutions are shown. We show that perturbations in derivative orders don't affect stability. Some conditions which FDEs must adhere to are presented.

When not specifically stated, the exposition follows that of Samko, Kilbas and Marichev [60]. If definitions and theorems are taken from books, page numbers are given after the bibliography number.

4.1 The Mittag-Leffler Function

The Mittag-Leffler function has several definitions, depending upon the number of input parameters.

The Mittag-Leffler function of one variable is defined as,

Definition 4.1.1 *([60] 21)*

$$E_\alpha(t) = \sum_{k=0}^{\infty} \frac{t^k}{\Gamma(\alpha k + 1)}, \alpha > 0. \tag{4.1}$$

The Mittag-Leffler function of two variables is considered to be a generalization of the exponential function.

Definition 4.1.2 *([60] 21)*

$$E_{\alpha,\beta}(t) = \sum_{k=0}^{\infty} \frac{t^k}{\Gamma(\alpha k + \beta)}, \alpha, \beta > 0. \tag{4.2}$$

Definitions (4.1.1) and (4.1.2) are linked by $E_\alpha(z) = E_{\alpha,1}(z)$. When $\alpha = \beta = 1$ we get the exponential function $E_{1,1}(z) = e^z$.

The multivariant Mittag-Leffler function is defined as,

Definition 4.1.3 *([56])*

$$E(\beta; t) := t^{\beta-1} E_{\mu-mu_1,\ldots,\mu-mu_v),\beta}(\lambda_1 t^{\mu-\mu_1}, \ldots, \lambda_v t^{\mu-\mu_v}), \beta > 0, \tag{4.3}$$

where

$$E_{(\alpha_1,\ldots,\alpha_v),\beta}(z_1, \ldots, z_v) := \sum_{k=0}^{\infty} \sum_{\substack{l_1 + \ldots + l_v = k \\ l_1 \geq, \ldots, l_v \geq 0}} (k; l_1, \ldots, l_v) \frac{\prod_{i=1}^{v} z_i^{l_i}}{\Gamma(\beta + \sum_{i=1}^{v} \alpha_i l_i)}$$

and

$$(k; l_1, \ldots . l_v) := \frac{k!}{l_1! \times \cdot \times l_v!}$$

are multinomial coefficients.

4.2 The Green's Function

Definition 4.2.1 *([56] 158)*

$$\begin{aligned} G(t) = \sum_{m=0}^{\infty} \frac{(-1)^m}{m!} & \sum_{k_0+k_1+\ldots+k_{n-2}=m, k_i \geq 0} (m; k_0, \ldots, k_{n-2}) \\ & \times \prod_{i=0}^{n-2} (-\lambda_{n-i})_i^k t^{(\alpha-\beta_1)m+\alpha+\sum_{j=0}^{n-2}(\beta_1-\beta_{n-j})k_j-1} \\ & \times E^{(m)}_{\alpha-\beta_1, \alpha+\sum_{j=0}^{n-2}(\beta_1-\beta_n-j)k_j}(\lambda_1 t^{\alpha-\beta_1}) \end{aligned}$$

where $(m; k_0, \ldots, k_{n-2}) = m! / \prod_{i=1}^{n-2}(k_i!)$ *is the multinomial coefficient and* $E^{(k)}$ *is the kth derivative of the Mittag-Leffler function with parameters* λ *and* μ*, given by*

$$E^{(k)}_{\lambda,\mu}(t) = \sum_{j=0}^{\infty} \frac{(j+k)! t^j}{j! \Gamma(\lambda j + \lambda k + \mu)}.$$

See Podlubny [56] for further details.

4.3 Hadamards Finite-Part Integral

Definition 4.3.1 *([60] 112)*

Let a function $\phi(t)$ be integrable on an interval $\epsilon < t < A$ for any $A > 0$ and $0 < \epsilon < A$. The function $\phi(t)$ is said to possess the Hadamard property at the point $t = 0$ if there exist constants a_k, b and $\lambda_k > 0$ such that

$$\int_{\epsilon}^{A} \phi(t)dt = \sum_{k=1}^{N} a_k \epsilon^{-\lambda_K} + b \ln \frac{1}{\epsilon} + J_0(\epsilon), \tag{4.4}$$

where $\lim_{\epsilon \to 0} J_0(\epsilon)$ exists and is finite. By definition

$$p.f. \int_{0}^{A} \phi(t)dt = \lim_{\epsilon \to 0} J_0(\epsilon). \tag{4.5}$$

The limit (4.5) is called a finite part (partie finie) of the divergent integral $\int_0^A \phi(t)dt$ in the Hadamard sense or simply an integral in the Hadamard sense. The constructive realization of the function $J_0(\epsilon)$ is sometimes called a regularization of the integral $\int_0^A \phi(t)dt$.

4.4 Analytical Solution of Fractional Differential Equations

The definition of the fractional differential contains a convolution integral. The properties of Laplace transforms can therefore be used to obtain analytical solutions for FDEs.

Miller and Ross [54] give the classical formula for the Laplace transform of the fractional derivative,

Definition 4.4.1 *([54] 121)*

$$\int_{0}^{\infty} e^{-st} {}_0D_t^{\alpha} f(t)dt = s^{\alpha}F(s) - \sum_{k=0}^{n-1} s^k [D^{\alpha-k-1} f(t)]_{t=0},$$

$$(n-1 < \alpha \leq n).$$

In the following section we show, using techniques borrowed from Miller and Ross [54], Podlubny [56] and Diethelm and Ford [14], how Laplace Transforms can be used to produce analytical solutions in terms of Mittag-Leffler and Green's functions, to single-term, multi-term and distributed order FDEs.

4.4.1 Analytical Solution of Single-term Fractional Differential Equations

Podlubny [56] shows us how to apply (4.4.1) to the example equation,

$$D^{1/2}y(t) - \lambda y(t) = 0, \quad (t > 0), \quad [D^{-1/2}f(t)]_{t=0} = C.$$

Applying the Laplace transform we obtain

$$F(s) = \frac{C}{s^{1/2} - \lambda}, \quad C = [{}_0D_t^{-1/2}f(t)]_{t=0}$$

and the inverse transform with the help of

$$\int_0^\infty e^{-pt}t^{\frac{k-1}{2}}E^{(k)}_{1/2,1/2}(\pm\lambda\sqrt{t})dt = \frac{k!}{(\sqrt{p} \mp \lambda)^{k+1}}, \quad (Re(p) > \lambda^2),$$

obtained from the Laplace transform of the Mittag-Leffler function in two parameters with $\alpha = \beta = 1/2$, gives

$$f(t) = Ct^{-1/2}E_{1/2,1/2}(\lambda\sqrt{t}).$$

More generally, consider the FDE,

$$D^\alpha y(t) - \lambda y(t) = g(t), \quad (t > 0), \quad [D^{\alpha-k}g(t)]_{t=0} = b_k, \quad (k = 1, 2, \ldots, n),$$

where $n - 1 < \alpha < n$.

Taking Laplace transforms gives,

$$s^\alpha Y(s) - \lambda Y(s) = H(s) + \sum_{k=1}^{n} b_k s^{k-1},$$

thus,

$$Y(s) = \frac{H(s)}{s^\alpha - \lambda} + \sum_{k=1}^{n} b_k \frac{s^{k-1}}{s^\alpha - \lambda}.$$

Taking inverse Laplace transforms using the general Laplace transform of the Mittag-Leffler function in two parameters,

$$\int_0^\infty e^{-pt}t^{\alpha k+\beta-1}E^{(k)}_{1/2,1/2}(\pm\lambda t^\alpha)dt = \frac{k!p^{\alpha-\beta}}{(p^\alpha \mp \lambda)^{k+1}}, \quad (Re(p) > |\lambda|^{1/\alpha})$$

gives,

$$y(t) = \sum_{k=1}^{n} b_k t^{\alpha-k}E_{\alpha,\alpha-k+1}(\lambda t^\alpha) + \int_0^t (t-\tau)^{\alpha-1}E_{\alpha,\alpha}(\lambda(t-\tau)^\alpha)h(\tau)d\tau.$$

4.4.2 Analytical Solution of Multi-term Fractional Differential Equations

Miller and Ross [54] exhibit an analytical solution to the inhomogeneous multi-term FDE. The solution is expressed in terms of the Green's function.

Theorem 4.4.2 *([54] 157)*

Let $g(t)$ be piecewise continuous on $C^2[0,t]$ and integrable and of exponential order on C. Let

$$[D^{nv} + a_1 D^{(n-1)v} + \ldots + a_n D^0]y(t) = g(t) \tag{4.6}$$

$$D^j y(0) = 0, \quad j = 0, 1, \ldots, N-1$$

be a fractional differential system of order (n,q), where N is the smallest integer greater than or equal to nv. Let

$$P(f) = g^t + a_1 x^{n-1} + \ldots + a_n$$

be the indicial polynomial and let

$$G(t) = L^{-1}\{P^{-1}(s^v)\}$$

be the fractional Green's function. Then

$$y(t) = \int_0^t G(t-\xi)x(\xi)d\xi$$

is the unique solution of (4.6).

4.4.3 Analytical Solution of Distributed Order Differential Equations

In Diethelm and Ford [14], the analytical solution of the distributed order differential equation

$$\int_a^b A(z)D^{m+z}y(t)dz = g(t), \tag{4.7}$$

first produced by Caputo in 1995, is stated as

$$y(t) = g(t) * L^{-1}\left(\frac{1}{p^m \int_a^b A(z)p^z dz}\right) + \sum_{n=0}^{m} t^n \frac{y^{(n)}(0)}{n!}. \tag{4.8}$$

4.5 Existence of Solutions

Diethelm and Ford [15] prove the existence of a solution to the single-term nonlinear FDE (2.8).

Theorem 4.5.1 *[15] Assume that* $D := [0, \chi^*] \times [y_0^{(0)} - a, y_0^{(0)} + a]$ *with some* $\chi^* > 0$ *and some* $a > 0$, *and let the function* $f : D \to \mathbb{R}$ *be continuous. Furthermore, define* $\chi := \min\{\chi^*, (a\Gamma(q+1)/||f||^\infty)^{1/q}\}$. *Then, there exists a function* $y : [0, \chi] \to \mathbb{R}$ *solving the initial value problem (2.8).*

Diethelm and Ford state that the generalization to multi-term equations is immediate.

4.6 Uniqueness of Solutions

Diethelm and Ford [15] prove the uniqueness of a solution to the single-term nonlinear FDE (2.8).

Theorem 4.6.1 *[15] Assume that* $D := [0, \chi^*] \times [y_0^{(0)} - a, y_0^{(0)} + a]$ *with some* $\chi^* > 0$ *and some* $a > 0$. *Furthermore, let the function* $f : D \to \mathbb{R}$ *be bounded on* D *and fulfil a Lipschitz condition with respect to the second variable, i.e.*

$$|f(x, y) - f(x, z)| \leq L|y - z|$$

with some constant $L > 0$ *independent of* x, y, *and* z. *Then, denoting* χ *as in theorem (4.5.1), there exists at most one function* $y : [0, \chi] \to \mathbb{R}$ *solving the initial value problem (2.8).*

Diethelm and Ford state that the generalization to multi-term equations is immediate.

4.7 Fractional Integration and Differentiation as Reciprocal Operators

For ordinary differential equations it is well known that integration and differentiation are reciprocal operators, if the former is applied first.

$$(d/dx) \int_0^x \varphi(t)dt = \varphi(x).$$

This is not the case for $\int_a^x \varphi'(t)dt$ however, due to the introduction of constants.

Similarly, for fractional operators, in the following theorem taken from Samko et al. [60] we show that fractional integration and differentiation are reciprocal operators, if the former is applied first.

Theorem 4.7.1 *([60] 43) In order that $f(x) \in I_{a+}^{\alpha}(L_1)$, $Re\ \alpha > 0$, it is necessary and sufficient that*

$$f_{n-\alpha}(x) =_{def} I_{a+}^{n-\alpha} f \in AC^n([a,b])$$

where $n = [Re\ \alpha] + 1$ and that

$$f_{n-\alpha}^{(k)}(a) = 0, k = 0, 1, 1, \ldots, n-1.$$

4.8 Perturbed Derivatives

4.8.1 Dependence on Parameters

In 'real world' applications, the differential order α would generally only be known to a certain degree of accuracy. Therefore, it is important to know whether small perturbations in the derivative adversely effect the accuracy of the solution.

In theorem 4.8.1 taken from Diethelm and Ford [16] we show that small perturbations in the derivative lead to small deviations in the solution of a linear single-term fractional differential equation

Theorem 4.8.1 *[16] Let $\alpha > 0, \epsilon > 0$ satisfy $m - 1 < \alpha - \epsilon < \alpha \leq m$ for some $m \in \{1, 2, 3\}$ and let y, z be the solutions, respectively, of the linear fractional differential equations*

$$D^{\alpha-\epsilon}(y - T_{m-1}[y])(t) = -y(t) + g(t), \quad y^{(k)}(0) = y_0^{(k)},$$

$$D^{\alpha}(z - T_{m-1}[z])(t) = -z(t) + g(t), \quad z^{(k)}(0) = y_0^{(k)},$$

with $k = 0, 1, \ldots, m-1$.

For $X < \infty$, we have

$$||y - z||_{L_\infty[0,X]} = 0(\epsilon).$$

Theorem 4.8.2 gives a more general result for non-linear FDEs.

Theorem 4.8.2 *[16] Assume that* $\mathbb{D} := [o, \chi^*] \times [y_0 - \alpha, y_0 + \alpha]$ *with some* $\chi^* > 0$ *and some* $\alpha > 0$. *Furthermore, let the function* $f : \mathbb{D} \to \mathbb{R}$ *be continuous and fulfill a Lipshitz condition with respect to the second variable. Moreover let* $\alpha > 0$ *and* $\delta > 0$ *such that* $m - 1 < \alpha - \delta < \alpha \leq m$. *Assume that* y *and* z *are the uniquely determined solutions of the initial value problems*

$$D^{\alpha-\delta}(y - T_{m-1}[y])(t) = f(t, y(t)), \quad y^{(k)}(0) = y^{(k)},$$

and

$$D^{\alpha}(z - T_{m-1}[z])(t) = f(t, z(t)), \quad z^{(k)}(0) = y^{(k)},$$

with $k = 0, 1, \ldots, m-1$, *respectively. Then we have the relation*

$$||y - z||_{\infty} = 0(\delta)$$

over any compact interval where both y *and* z *exist.*

In theorem 4.8.3 we show a corresponding solution for multi-term equations.

Theorem 4.8.3 *[15] Let* y *be the solution of*

$$D_*^{\alpha} y(t) = f(t, y(t), D_*^{\beta_1} y(t), D_*^{\beta_2} y(t), \ldots, D_*^{\beta_n} y(t))$$

with initial conditions

$$y^{(k)}(0) = y_0^{(k)}, k = 0, 1, \ldots, \lfloor \alpha \rfloor$$

and let z *be the solution of*

$$D_*^{\tilde{\alpha}} z(t) = f(t, z(t), D_*^{\tilde{\beta}_1} z(t), D_*^{\tilde{\beta}_2} z(t), \ldots, D_*^{\tilde{\beta}_n} z(t))$$

with initial conditions

$$z^{(k)}(0) = y_0^{(k)}, k = 0, 1, \ldots, \lfloor \alpha \rfloor$$

where $|\alpha - \tilde{\alpha}| < \epsilon, |\beta_j - \tilde{\beta}_j| < \epsilon$. *For* $T < \infty$, *we have*

$$||y - z||_{L_\infty[0,t]} = O(\epsilon), \epsilon \to 0.$$

4.9 Absolute Continuity

Definition 4.9.1 *([60] 2). A function* $f(x)$ *is called absolutely continuous on an interval* Ω, *if for any* $\epsilon > 0$ *there exists a* $\delta > 0$ *such that for any finite set of pairwise nonintersecting intervals* $[a_k, b_k] \subset \Omega$, $k = 1, 2, \ldots, n$, *such that* $\sum_{k=1}^{n} (b_k - a_k) < \delta$, *the inequality* $\sum_{k=1}^{n} |f(b_k) - f(a_k)| < \epsilon$ *holds. The space of those functions is known as* $AC(\Omega)$.

4.10 Conclusions

In this chapter we have stated enough of the relevant mathematics so as to make this thesis self contained.

The next chapter begins our exposition of the numerical methods used to solve FDEs. We discuss 5 methods for the solution of single-term FDEs. In chapter 6 we compare these methods for numerical efficiency.

Chapter 5

Single-term FDE Methods

In this chapter we introduce several methods for the numerical solution of single-term fractional differential equations (FDEs). More detail on some of the methods, speed up algorithms and additional methods can be found in [23], [30], [31], [57] and [63].

We are interested in numerical solutions of single-term FDEs of the general form

$$D^{\alpha}y(t) = f(t, y(t)), \tag{5.1}$$

where $0 < \alpha < m, m = \lceil \alpha \rceil$ subject to the initial conditions

$$y^{(k)}(0) = y_0^{(k)}, k = 0, \ldots, m-1.$$

Most of the methods that we consider are more naturally applied in the context of linear equations, for which

$$f(t, y(t)) = \lambda y(t) + g(t), \tag{5.2}$$

and it is with respect to equations of this form that we shall perform various numerical experiments and comparisons.

In chapter 6 we will discuss two cases:

Case A. $0 < \alpha < 1$, subject to the initial condition

$$y(0) = y_0.$$

Case B. $1 < \alpha < m$, subject to the initial conditions

$$y^{(k)}(0) = y_0^{(k)}, k = 0, \ldots, m-1.$$

We choose to compare the merits of 5 methods:

1. Implicit quadrature, K. Diethelm [11]
2. Predictor corrector, K. Diethelm, N. J. Ford and A. D. Freed [19]
3. Approximate Mittag-Leffler, K. Diethelm and Y. Luchko [24]
4. Collocation, L. Blank [2]
5. Finite differences, R. Gorenflo [33]

For case **A** all 5 methods are compared, however for case **B** the implicit quadrature scheme is not viable and shall not be considered, for reasons discussed later (see section 5.1).

The pseudo-code implementation of methods 1-3 can be found in [21]. All Matlab implementations will be available from

`www.chester.ac.uk/mathematics`

Method 1 is based on the implicit trapezoidal quadrature formulae. Method 2 is a fractional version of an Adams Moulton $(P(EC)^M C)$ predict correct evaluate correct algorithm, where M is the number of corrector iterations. Diethelm and Luchko express the solution of a FDE as a sum of Mittag-Leffler functions. They then use a convolution quadrature to approximate this analytical solution. Blank uses collocation techniques and Gorenflo uses a finite difference approach.

The most important methods (in regard to this thesis) are the implicit quadrature and the predictor corrector methods. We shall therefore give these more attention and shall just state the algorithms of the other methods.

We first introduce some notation that we shall use for each numerical scheme. We set a fixed step length $h > 0$ and use y_j as a representation of the approximation of $y(jh)$ at integer multiples of the step length. An equispaced grid $t_j = j/n$ is set.

5.1 Implicit Quadrature, K. Diethelm [11]

Diethelm [11] introduces a numerical scheme which appears to use the Riemann-Liouville fractional derivative in defining a backward difference formula generalization. Upon closer inspection, by incorporating the initial condition $y(0) = y_0$ into the equation, the problem has been transformed into a fractional differential

equation with Caputo derivatives (see section 2.4). Thus any complication when inhomogeneous initial conditions arise is removed.

By interchanging differentiation and integration of the Riemann-Liouville fractional derivative (see definition 2.2.2) we obtain,

$$D^{\alpha}y(t) = \frac{1}{\Gamma(-\alpha)}\int_0^t \frac{y(u)}{(t-u)^{\alpha+1}}du, \tag{5.3}$$

where the integral is interpreted in a Hadamard finite-part sense (see Section 4.3).

Applying our approximation to the equispaced grid for $j = 1, 2, \ldots, n$, we obtain,

$$\begin{aligned} g(t_j) + \lambda y(t_j) &= \frac{1}{\Gamma(-\alpha)}\int_0^{t_j} \frac{y(u) - y(0)}{(t_j - u)^{\alpha+1}}du \\ &= \frac{t_j^{-\alpha}}{\Gamma(-\alpha)}\int_0^1 \frac{y(t_j - t_j w) - y(0)}{w^{\alpha+1}}dw. \end{aligned}$$

Replacing the integral by a compound quadrature formula [12], with equispaced nodes $0, 1/j, 2/j, \ldots, 1$ for each j, gives

$$Q_j[v] = \sum_{i=0}^{j} w_{ij} v(i/j) \approx \int_0^1 v(u)u^{-\alpha-1}du,$$

with remainder term

$$R_j[v] = \int_0^1 v(u)u^{-\alpha-1}du - Q_j[v].$$

The following implicit formula gives Diethelm's numerical scheme for the solution of the equation (5.1) with the linear condition (5.2):

$$y_j = \frac{1}{w_{0j} - (j/n)^{\alpha}\Gamma(-\alpha)\lambda}\left(\left(\frac{j}{n}\right)^{\alpha}\Gamma(-\alpha)\, g(t_j) - \sum_{i=1}^{j} w_{ij} y_{j-i} - \frac{1}{\alpha}y_0\right), \tag{5.4}$$

where weights w_{ij} are,

$$\alpha(1-\alpha)j^{-\alpha}w_{ij} = \begin{cases} -1 & \text{for } i = 0, \\ 2i^{1-\alpha} - (i-1)^{1-\alpha} - (i+1)^{1-\alpha} & \text{for } i = 1, 2, \ldots, \text{j-1}, \\ (\alpha-1)i^{-\alpha} - (i-1)^{1-\alpha} + i^{1-\alpha} & \text{for } i = j. \end{cases}$$

Diethelm states that when using functions that are sufficiently smooth the error behaves as $O(h^{2-\alpha})$. The conditions to achieve this are not explicitly stated. The method is analysed for $0 < \alpha < 1$. Diethelm states that the extension to $1 < \alpha < 2$ should present no major difficulty, as this would reduce the order of convergence to under $O(h)$ this case will not be studied further.

5.2 Predictor Corrector, K. Diethelm, N. J. Ford and A. D. Freed [19]

Diethelm et al. [19] introduce an algorithm of $P(EC)^M E$ (predict evaluate correct evaluate) type, for the solution of the non-linear FDE,

$$D^\alpha y(t) = f(t, y(t)). \quad 0 \leq t \leq 1, \tag{5.5}$$

where $0 < \alpha < m, m = \lceil \alpha \rceil$ subject to the initial conditions,

$$y^{(k)}(0) = y_0^{(k)}, k = 0, \ldots, m-1.$$

The algorithm can be used to solve equation (5.1), in the particular case (5.2). This enables us to compare the computational efficiency of the $P(EC)^M E$ algorithm directly with the other single-term FDE methods.

The main section of the algorithm is of an explicit nature, a predictor being used to start the algorithm. Repeated corrections and evaluations can then be applied to achieve optimum efficiency.

Equation (5.5) is equivalent to the Volterra integral equation (see [16] Lemma 2.1)

$$y(t) = \sum_{k=0}^{m-1} \frac{t^k}{k!} y_0^{(k)} + \frac{1}{\Gamma(\alpha)} \int_0^t (t-u)^{\alpha-1} f(u, y(u)) du. \tag{5.6}$$

We can replace the integral in equation (5.6) by the product rectangle rule

$$\int_0^{t_{j+1}} (t_{j+1} - u)^{\alpha-1} f(u) du \approx \sum_{i=0}^{j} b_{i,j+1} f(t_i)$$

where,

$$b_{i,j+1} = \frac{h^\alpha}{\alpha}((j+1-i)^\alpha - (j-i)^\alpha). \tag{5.7}$$

Thus the predictor y_{j+1}^P is determined by the fractional Adams-Bashforth method,

$$y_{j+1}^P = \sum_{k=0}^{m-1} \frac{t_{j+1}^k}{k!} y_0^{(k)} + \frac{1}{\Gamma(\alpha)} \sum_{i=0}^{j} b_{i,j+1} f(t_i, y_i), \tag{5.8}$$

with weights (5.7).

The corrector formula is the fractional version of the Adams-Moulton method, which can be one or multi-step. Using standard results from quadrature theory (cf. [23]), the integral in equation (5.6) can be approximated by

$$\int_0^{t_{j+1}} (t_{j+1} - u)^{\alpha-1} f(u) du \approx \frac{h^\alpha}{\alpha(\alpha+1)} \sum_{i=0}^{j+1} a_{i,j+1} f(t_i),$$

where

$$a_{i,j+1} = \begin{cases} (j^{\alpha+1} - (j-\alpha)(j+1)^\alpha) & \text{if } i = 0, \\ ((j-i+2)^{\alpha+1} + (j-i)^{\alpha+1} - 2(j-i+1)^{\alpha+1}) & \text{if } 1 \le i \le j, \\ 1 & \text{if } i = j+1. \end{cases} \tag{5.9}$$

Thus the corrector y_{j+1} is determined by

$$y_{j+1} = \sum_{k=0}^{m-1} \frac{t_{j+1}^k}{k!} y_0^{(k)} + \frac{1}{\Gamma(\alpha)} \left(\sum_{i=0}^{j} a_{i,j+1} f(t_i, y_i) + a_{j+1,j+1} f(t_{i+1}, y_{i+1}^P) \right), \tag{5.10}$$

with weights (5.9).

An approximate solution to equation (5.1) with either a non-linear function $f(t, y(t))$ or a linear function $\lambda y(t) + g(t)$, can be obtained using the predictor (5.8) and the corrector (5.10) with the relevant weights.

5.2.1 Predictor Corrector - Multiple Corrector Iterations

The predictor corrector's order of convergence depends upon the predictor's order of convergence . In this case the predictor is a fractional generalization of the Euler method, and it is widely known that the error of the Euler method behaves as $O(h)$. The error of the Adams-Bashford algorithm is well known to be $O(h^2)$. The predictor corrector algorithm's order of convergence can be increased by performing additional corrector iterations. This makes the fractional $P(EC)^M E$ method's error behave closer to that of the Adams-Bashford method.

When $\alpha < 1$ and no additional corrector iterations are applied, Diethelm and Ford [20] proved that the error of the fractional predictor corrector algorithm behaves as

$$\max_{j=0,1,\ldots,n} |y(t_j) - y_j| = O(h^p),$$

where $p = \min(2, 1 + \alpha)$.

If M additional corrector iterations are applied the convergence order increases to $O(h^2)$. The number of corrector iterations needed to achieve this depends upon $|y^P(t_{j+1}) - y_{j+1})|$, which is the error due to the predictor. This error, for any particular h, decays linearly with increasing M, and therefore it may be possible to determine the number of corrector steps required to achieve $O(h^2)$. As numerical complexity increases, with increasing M, the issue becomes, at what point does increasing M become numerically inefficient to do? This problem will be addressed later.

5.3 Approximate Mittag-Leffler, K. Diethelm and Y. Luchko [24]

Diethelm and Luchko use the observation that a fractional differential equation has an exact solution, which can be expressed as a Mittag-Leffler type function. They then use convolution quadrature [12] and discretized operational calculus to produce an approximation to this Mittag-Leffler function.

The numerical scheme for the solution of equations of the form (5.1) with a linear function as in (5.2), is given by,

$$y_j = \tilde{y}_j + \sum_{i=1}^{N} \hat{y}_j. \tag{5.11}$$

This is achieved by considering two cases, the homogeneous problem (i.e. $g = 0$) with zero initial conditions, and the inhomogeneous problem (i.e. $g \neq 0$) with non-zero initial conditions.

HOMOGENEOUS:

$$\tilde{y}_j := \sum_{k=0}^{m-1} y_0^{(k)} \left(\frac{(t_j)^k}{k!} + u_h(k; t_j) \right), j = 1, \ldots, n, \tag{5.12}$$

where

$$u_h(k; t_j) := \omega_j(k; h)/h.$$

Here $\omega_i(k;h), k = 0, 1, \ldots, m-1, i = 0, 1, \ldots, n$ are the coefficients of the power series

$$F_k(\delta(\zeta)/h) = \sum_{i=0}^{\infty} \omega_i(k;h)\zeta^i, \tag{5.13}$$

and $\delta(\zeta) = \sum_{i=0}^{\infty} \delta_i \zeta^i$ is the quotient of the generating polynomials of a linear multistep method,

$$F_k(s) := \frac{\lambda s^{\alpha-k-1}}{s^\alpha - \lambda s^\alpha},$$

and the natural numbers $l_k, k = 0, \ldots, m-1$ are determined from the condition

$$\begin{cases} m_{l_k} \geq k+1 \\ m_{l_k+1} \leq k \end{cases}.$$

INHOMOGENEOUS:

Luchko and Gorenflo [51] showed that the analytical solution $\tilde{y}$ of equation (5.1) could be represented as

$$\tilde{y} = \int_0^t E(\mu;t)g(t-\tau)d\tau. \tag{5.14}$$

Then using the numerical scheme introduced in Lubich [50], the convolution integral (5.14) at the point t_j, is approximated by,

$$\tilde{y}_j = \sum_{0 \leq i \leq j} \omega_j(h)g(t_{j-i}), j = 1, \ldots n. \tag{5.15}$$

Where $\omega_i(h)$ is given by

$$F(\alpha;\delta(\zeta)/h) = \sum_{i=0}^{\infty} \omega_i(h)\zeta^i.$$

A simple modification obtains the required convergence order, thus the inhomogeneous case is given by,

$$\tilde{y}_j = \hat{y}_j + \sum_{i=i_0}^{q-1} w_{ji}(h)g(t_i) \quad , q-1 < p-\gamma \leq q \in \mathbb{N}. \tag{5.16}$$

Where the correction quadrature weights $w_{ji}(h)$, $i = i_0, \ldots, q-1$ are determined from the Vandermonde system.

Diethelm and Luchko's method can have order of convergence $O(h)$ or $O(h^2)$, depending on the order of the underlying quadrature method. $p = 1$, represents

a method based upon Euler's rule. $p = 2$ represents a method based upon a trapezoidal rule.

Lubich [50] proves that higher order methods are also available, however Diethelm et al. [13] show that while this higher convergence is theoretically possible, implementation issues place considerable doubt on the efficiency and reliability of all multi-step methods where $p > 2$. Throughout this thesis we therefore use only linear multistep methods with $p = 1, 2$.

We let the exact analytical solution, for a fixed interval T be $y(T)$ and the approximation at T using n step lengths be $y_n(T)$. Here, and subsequently, EOC, represents the experimental order of convergence evaluated using the formula

$$EOC = log_2 \left(\frac{\mid y(T) - y_n(T) \mid}{\mid y(T) - y_{2n}(T) \mid} \right). \tag{5.17}$$

All time values are given in seconds. KFlops = 1000× Flops.

For table 5.1 we use the test equation,

$$D^{1/2}y + y = t^2 + \frac{2t^{3/2}}{\Gamma(5/2)} \tag{5.18}$$

with

$$y(0) = 0.$$

	p=1				p=2			
h	error y(1)	KFlops	Time	EOC	error y(1)	KFlops	Time	EOC
1/8	3.43e-02	5.7	0.00		1.90e-03	5.9	0.00	
1/16	1.73e-02	21.4	0.00	0.99	5.00e-04	22.3	0.00	1.93
1/32	8.66e-03	89.7	0.05	1.00	1.23e-04	92.9	0.05	1.96
1/64	4.34e-03	395.8	0.11	1.00	3.29e-05	404.8	0.11	1.97
1/128	2.17e-03	1758.4	0.22	1.00	8.40e-06	1797.9	0.22	1.98

Table 5.1: How the order of the convolution quadrature method effects convergence

It can be seen from table 5.1 that convergence increases by an order of magnitude while the order of magnitude of the work done remains constant. We shall therefore use the $p = 2$ method throughout the rest of this thesis.

5.4 Collocation, L. Blank [2]

Blank uses the Riemann-Liouville definition of the fractional derivative D_*. However, by subtracting the sum

$$\sum_{k=0}^{m-1} \frac{1}{k!} t^k y^{(k)}(0)$$

inside the derivative, the Riemann-Liouville definition becomes equivalent to the Caputo definition. i.e.

$$D^\alpha y(t) = D_*^\alpha \left(y(t) - \sum_{k=0}^{m-1} \frac{1}{k!} t^k y^{(k)}(0) \right) = \lambda y(t) + g(t), \tag{5.19}$$

with $m-1 < \alpha \leq m, m \in \mathbb{N}$, and initial conditions $y^{(k)}(0)(k = 0, \ldots, m-1)$.

Assuming an equidistant mesh $t_j = j.h$ and a r-times globally continuously differentiable polynomial spline solution u, a collocation method with polynomial splines is implemented. For the reader unfamiliar with collocation techniques, a spline on each subinterval $[t_j, t_{j+1}]$ is described by,

$$u(t_j + vh) = \sum_{l=0}^{s+r} a_l^{(j)} v^l, \quad v \in [0, 1],$$

where s is the number of collocation parameters, then u is found to satisfy (5.1) in the collocation points $t_{j,i} = t_j + c_i h$ for $i = 1, \ldots, s$. i.e.

$$D^\alpha \left(u(t_j, i) - \sum_{k=0}^{m-1} \frac{1}{k!} t_{j,i}^k y^{(k)}(0) \right) = \lambda u(t_{j,i}) + g(t_{j,i}) \tag{5.20}$$

It was shown by Blank that if (m-1)-times continuously differentiable splines (i.e. $r = m-1$) were chosen, the above approximation can be portrayed as,

$$\left[D^\alpha \left(u(t) - \sum_{k=0}^{m-1} \frac{1}{k!} t^k y^{(k)}(0) \right) |_{t=t_{j,i}} \right]_{i=1,\ldots,s} = \frac{h^{-\alpha}}{\Gamma(1-\alpha)} \sum_{i=0}^{j} V^{(j-i)} b^{(i)} \tag{5.21}$$

with suitable weight matrices $V^{(i)}$.

Blank [2] theorem 4 states that the numerical solution u of (5.20), given by the collocation approximation is determined by the following systems of equations.

On the first subinterval $[0, t_1]$,

$$a^{(0)} = \begin{pmatrix} \ddots & & 0 \\ & h^k \frac{1}{k!} & \\ 0 & & \ddots \end{pmatrix}_{k=0,\ldots,m-1} \left(y^{(k)}(0)\right)_{k=0,\ldots,m-1}$$

$$b^{(0)} = INV\left\{\lambda C a^{(0)} + g^{(0)}\right\} \tag{5.22}$$

while on $[t_j, t_{j+1}]$ for $j \geq 1$,

$$a^{(j)} = Diff \begin{pmatrix} a^{(j-1)} \\ b^{(j-1)} \end{pmatrix}$$

$$b^{(j)} = INV\left(\lambda^{1/\alpha} C a^{(j)} + g^{(j)} - \frac{h^{-\alpha}}{\Gamma(1-\alpha)} \sum_{i=0}^{j-1} V^{(j-i)} b^{(i)}\right).$$

With weight matrices $V^{(j)}$, defined by,

$$K^{(j)} = \left((j + c_i)^{m-1+l-\alpha}\right)_{\substack{i=1,\ldots,s \\ l=1,\ldots,s}}$$

$$P_{k,l} = \begin{cases} \frac{(l+m-1)!}{(l-k)!} \Pi_{p=1}^{m-1+k} \frac{1}{p-\alpha} & \text{for } k \leq l \\ 0 & \text{for } k > l \end{cases}$$

This yields the equations

$$\begin{aligned} V^{(0)} &= K^{(0)} P_{diag} \\ V^{(j)} &= K^{(j)} P_{diag} - K^{(j-1)} P, \qquad (j \geq 1) \end{aligned}$$

also,

$$\begin{aligned} g^{(j)} &= (g(t_j + c_i h)_{i=1,\ldots,s}, \\ Diff &= (Diff_{k,l})_{\substack{k=1,\ldots,s \\ l=1,\ldots,s}} \end{aligned} \tag{5.23}$$

with,

$$Diff_{k,l} = \begin{cases} \frac{l!}{k!(l-k)!} & \text{for } k \leq l \\ 0 & \text{for } k > l \end{cases} \tag{5.24}$$

and,

$$INV = \left[\frac{h^{-\alpha}}{\Gamma(1-\alpha)} K^{(0)} P_{diag} - \lambda (c_i^{m-1+l})_{\substack{i=1,\ldots,s \\ l=1,\ldots,s}}\right]^{-1}. \tag{5.25}$$

5.4.1 Choice of Collocation Points for Blank's Method

Blank's goal was to reflect the qualitative behavior accurately and therefore high convergence was not a priority. For table 5.2 we use test equation (5.18) with the given initial conditions. It can be seen from table 5.2 that choice and

h	Collocation points 0.1, 0.5, 1.0				Collocation points 0.4, 0.5, 1.0			
	error y(1)	KFlops	Time	EOC	error y(1)	KFlops	Time	EOC
1/8	1.06e-01	5.2	0.06		6.14e-02	5.2	0.05	
1/16	5.24e-02	11.2	0.11	1.02	3.38e-02	11.2	0.11	0.86
1/32	2.56e-02	27.8	0.16	1.04	1.81e-02	27.8	0.17	0.90
1/64	1.25e-02	79.5	0.33	1.03	9.54e-03	79.5	0.39	0.93
1/128	6.13e-03	256.6	0.93	1.03	4.95e-03	256.6	1.10	0.95
	Collocation points 0.1, 0.5, 1.0				Collocation points 0.4, 0.5, 1.0			
1/8	3.31e-04	8.6	0.05		1.25e-01	13.4	0.06	
1/16	3.54e-04	18.4	0.11	-0.10	6.07e-02	28.3	0.11	1.04
1/32	2.65e-04	45.5	0.22	0.42	2.94e-02	69.4	0.22	1.05
1/64	1.83e-04	129.7	0.50	0.54	1.42e-02	196.0	0.60	1.04
1/128	1.16e-04	417.9	1.32	0.66	6.94e-03	627.5	1.32	1.04

Table 5.2: Effect of collocation point change

number of collocation parameters does not affect the order of the method in the limit as $h \to 0$. It does affect the absolute error at particular step sizes, it simultaneously affects the convergence rate in an adverse manner, thus for small h the difference evens out. If the user wishes to use larger step sizes, carefully chosen collocation points could reduce the absolute error. It should be noted that additional collocation points do not necessarily reduce absolute error.

5.5 Finite Differences, R. Gorenflo [33]

The finite difference approach to fractional integration can be attributed to Grünwald and Letnikov. Podlubny [56] uses the first order method (accuracy $O(h)$ and confirms a convergence of $O(h)$. Gorenflo [33] summarizes this method and introduces a second order difference method (accuracy $O(h^2)$). He also states sufficiently smooth zero extension conditions to achieve the desired accuracy. The conditions on the second order method are restrictive, which make the method of little use, we will not explore it further.

derivative, the approximation of the α-th derivative is

$$h^{-\alpha} \nabla_h^{\alpha} y(t) = \tilde{D}^{\alpha} y(t) = h^{-\alpha} \sum_{i=0}^{[t/h]} (-1)^i \binom{\alpha}{i} y(t - ih). \tag{5.26}$$

Thus the first order approximation to the single term problem (5.1) can be represented by the difference scheme,

$$y_j = -\lambda h^{\alpha} y_{j-1} - \sum_{i=1}^{j} w_i^{(\alpha)} y(t_{j-i}) + h^{\alpha} g(t_j) \tag{5.27}$$

where,

$$w_i^{(\alpha)} = (-1)^i \binom{\alpha}{i}.$$

5.6 Conclusions

In this chapter we have introduced five single-term methods, all of which are suitable for linear FDEs, but only the predictor corrector method is suitable for non-linear FDEs.

Experimentation suggests that the number of collocation parameters in the Blank method does not affect the convergence order, but does affect absolute error. The second order Diethelm and Luchko scheme $p = 2$ was found to have a comparable workload to the single order scheme $p = 1$.

In the next chapter we shall compare the numerical efficiency of the five methods for various test equations. We shall introduce a new graphical technique, which displays relative numerical efficiency of FDE methods and shows how the optimum linear FDE method depends upon the order of the derivative α and the function $g(t)$.

Chapter 6

Numerical Comparison of Single-term Methods

6.1 How Should we Judge a Good Numerical Method?

The effectiveness of numerical methods is classically considered using ideas such as the convergence, consistency and stability of the method. Of these the convergence order of the method is regarded as paramount. The order of a method describes the rate at which an error will decrease, over a fixed interval $[0, T]$, as a result of decreasing the step length. We talk about a method being of order $O(h)$, if the error halves for a halving step length, while a method would be of order $O(h^2)$ if the error decreases by a factor of 4 when the step length is halved. These orders are for asymptotic results and apply only as $h \to 0$ so for finite step lengths we may observe these orders being exceeded. As a further observation one needs to understand that the order of the method gives an error bound over some fixed time interval. There is a constant that depends on the method and the particular equation that determines the size of the actual error. Therefore two methods with the same order could have extremely different actual errors when they are applied

Alongside the effectiveness of the numerical method is a requirement to keep the computational cost low. Typically the user of an algorithm would wish for a solution with a certain error tolerance, therefore the computational cost would be the time required for a computer to achieve this. This is of great importance as sometimes results are available too late to be of use.

Computational cost can be measured either, in the number of FLOPS (floating point operations) or the actual run time required for a calculation. Each has its

advantages and disadvantages. Measuring with time depends on the loading of the computer system, and therefore may vary from run to run. FLOP counts are consistent for each run, but may not allow a direct conversion to estimate run time. Throughout this thesis we choose to mainly use FLOPS as our method of counting, this enables any person wishing compare different algorithms with the ones in this thesis to replicate our results. We talk about a method having a computational cost $O(n^p)$, where n is the number of steps, to show the effect of changing $n = \frac{1}{h}$ on the work required by the algorithm. For further information consult [38], [59] or [41].

In the past surprisingly little work [27] has been done to try and calculate the true *efficiency* of numerical methods.

The order of convergence and computational cost of a method often conflict with each other, because the 'rate of increase in computational cost' is often greater with higher order methods. There is also a battle between the order of a method and the magnitude of any constants involved. Therefore a lower order method may be more economical than a method of higher order for large step sizes (small number of steps), before the dominance of the superior convergence order method emerges. In section 6.3 a graphical representation of these properties is introduced. This enables the designer of a numerical method to directly compare their method with any alternatives available. The method can also be used in conjunction with small run times, of several methods, to predict which method is most economical over large time intervals, for any given equation.

For each method, the technique involves plotting the 'log of error' against the 'log of FLOP count', for a series of decreasing step lengths. This produces a line for each method, which we then extrapolate. Where the lines depicting each method cross, representing a critical point of efficiency, the line with the lower gradient at this point is the most efficient. We show that the step size where this occurs depends upon the value α and the function $g(t)$. This new graphical representation allows a predictive computative tool to be created, which obtains the best method for any particular FDE.

All calculations were performed on a 500MHz Pentium based laptop in double precision arithmetic.

6.2 Choice of Test Equations

The choice of test equations is limited to where the Caputo derivative of the known solution $y(t)$ is integrable. Thus the main class of test equations have solutions which are polynomials. The trigonometric and exponential functions are of some use, however, for certain values of α the Caputo derivative of the

known solution $y(t)$ is not easily integrable. The problem can be avoided by using the function's series expansions. These expansions also help illustrate the way in which the differentiability of the Caputo derivative of the known solution $y(t)$, effects the manner in which the numerical methods behave relative to one another.

Equations can possess homogeneous or inhomogeneous initial conditions. This does not influence the behaviour of the numerical methods, but can effect the nature of the test equations. We therefore, for simplicity, just use homogeneous equations.

In [24] Diethelm and Luchko examine the convergence behaviour of the implicit quadrature, predictor corrector and approximate Mittag-Leffler algorithms over various smoothness assumptions. We shall not repeat these results here. Our aim is to demonstrate the relative numerical efficiency of the algorithms.

6.2.1 Linear Equations

In this section we will be concerned with solutions to the linear single-term FDE

$$D^{\alpha}y + \lambda y = g(t), \tag{6.1}$$

for 2 special cases,

A. $0 < \alpha < 1$, subject to the initial condition

$$y(0) = y_0.$$

B. $1 < \alpha < m$, subject to the initial conditions

$$y^{(k)}(0) = y_0^{(k)}, k = 0, \ldots, m-1.$$

Case A

To illustrate the convergence behaviour of the various single-term FDE methods we vary the value of $\alpha < 1$ of the test equation,

$$D^{\alpha}y + y = t^2 + \frac{2t^{2-\alpha}}{\Gamma(3-\alpha)}$$

$y(0) = 0$, exact solution $y = t^2$, for $\alpha = 1/4, 1/2, 3/4$. These are chosen to highlight how the convergence order can depend upon the value of α. In [11] and [19] Diethelm and Ford prove that the convergence of the implicit quadrature and

predictor corrector methods are of the order $O(h^{2-\alpha})$ and $O(h^{1+\alpha})$ respectively. The approximate Mittag-Leffler method [24] (with $p = 2$) is of order the $O(h^2)$. The finite differences [33] method is of order $O(h)$. We are not aware of a published proof on the convergence order of the Blank collocation method [2]. We have found $O(h)$ from our experimental data.

In tables 6.1 to 6.6, we support these results with numerical experiments.

From an engineer's prospective however, what is of more interest, is the duration of time taken to achieve a solution of a particular accuracy. Tables 6.1 to 6.6, portray the information required to assess this, which is often not immediately obvious. For example, in table 6.1, the approximate Mittag-Leffler method with $h = 1/128$, has an absolute error of 4.71e-06, compared to the absolute error of 2.73e-05 for the implicit quadrature method, however the number of KFlops required (1796.4 compared to 42.1) is much greater for the approximate Mittag-Leffler method.

It turns out that neither of these methods is the most numerically efficient because, if we look at table 6.2, when we take into account the size of error and the KFlop count, we can see that the predictor corrector method with ($M = 8$) is best. This problem is addressed in section 6.3 by introducing a new graphical technique.

Note, in table 6.6 the EOC from $h = 1/8 \rightarrow 1/16$ of the predictor corrector (M=2) method has a value of 4.49. This would appear to be unusual. Due to the structure of the algorithm for large step sizes and a small number of corrector iterations, error values take time to propegate through the solution, so EOC values appear unusually large. In table 6.1 and thoughout the rest of this thesis, all calculations are given to 3 significant figures, however the accual EOC value has been calculated using a higher accuracy.

h	Predictor corrector (M=1)				Implicit quadrature			
	error y(1)	KFlops	Time	EOC	error y(1)	KFlops	Time	EOC
1/8	9.84e-02	2.5	0.05		2.88e-03	1.2	0.00	
1/16	3.50e-02	5.2	0.11	1.49	9.15e-04	2.6	0.00	1.65
1/32	1.27e-02	11.5	0.16	1.46	2.87e-04	5.9	0.05	1.68
1/64	4.72e-03	27.0	0.28	1.43	8.88e-05	14.9	0.11	1.69
1/128	1.79e-03	70.4	0.55	1.40	2.73e-05	42.1	0.22	1.70
	Approximate Mittag-Leffler				Collocation points 0.1, 0.5, 1.0			
1/8	1.19e-03	5.9	0.06		1.25e-01	5.2	0.05	
1/16	2.99e-04	22.3	0.06	1.99	6.24e-02	11.2	0.11	1.00
1/32	7.50e-05	92.9	0.06	2.00	3.08e-02	27.8	0.17	1.02
1/64	1.88e-05	405.9	0.11	2.00	1.51e-02	79.5	0.33	1.03
1/128	4.71e-06	1796.4	0.27	2.00	7.40e-03	256.6	0.93	1.03
	Finite differences							
1/8	1.63e-02	0.9	0.00					
1/16	8.19e-03	2.3	0.05	0.99				
1/32	4.11e-03	6.9	0.06	1.00				
1/64	2.06e-03	23.1	0.11	1.00				
1/128	1.03e-03	83.0	0.44	1.00				

Table 6.1: Test equation $D^{1/4}y + y = t^2 + \frac{2t^{1.75}}{\Gamma(2.75)}$

h	Predictor corrector (M=2)				Predictor corrector (M=4)			
	error y(1)	KFlops	Time	EOC	error y(1)	KFlops	Time	EOC
1/8	3.34e-02	1.9	0.06		9.39e-03	2.9	0.05	
1/16	1.14e-02	4.0	0.06	1.56	2.16e-03	6.4	0.11	2.12
1/32	3.81e-03	9.6	0.11	1.58	4.93e-04	15.4	0.17	2.13
1/64	1.27e-03	25.2	0.22	1.58	1.13e-04	40.9	0.39	2.13
1/128	4.25e-04	74.9	0.44	1.58	2.59e-05	122.8	0.77	2.12
	Predictor corrector (M=8)				Predictor corrector (M=16)			
1/8	1.06e-03	5.1	0.06		1.06e-03	9.3	0.16	
1/16	2.68e-04	11.2	0.16	1.98	2.68e-04	20.8	0.33	1.98
1/32	6.78e-05	27.0	0.33	1.98	6.78e-05	50.3	0.66	1.99
1/64	1.71e-05	72.4	0.66	1.99	1.71e-05	135.4	1.26	1.99
1/128	4.29e-06	218.5	1.37	1.99	4.29e-06	410.0	2.53	1.99

Table 6.2: Test equation $D^{1/4}y + y = t^2 + \frac{2t^{1.75}}{\Gamma(2.75)}$

h	Predictor corrector (M=1)				Implicit quadrature			
	error y(1)	KFlops	Time	EOC	error y(1)	KFlops	Time	EOC
1/8	2.97e-02	1.3	0.00		1.07e-02	1.2	0.00	
1/16	9.18e-03	2.8	0.05	1.69	3.90e-03	2.6	0.05	1.45
1/32	2.94e-03	6.6	0.05	1.64	1.41e-03	5.9	0.06	1.47
1/64	9.71e-04	17.3	0.16	1.60	5.07e-04	14.9	0.11	1.48
1/128	3.27e-04	51.0	0.33	1.57	1.81e-04	42.1	0.22	1.49
	Approximate Mittag-Leffler				Collocation points 0.1, 0.5, 1.0			
1/8	1.90e-03	5.9	0.06		1.06e-01	5.2	0.06	
1/16	5.00e-04	22.3	0.06	1.99	5.24e-02	11.2	0.11	1.02
1/32	1.29e-04	92.9	0.06	2.00	2.56e-02	27.8	0.16	1.04
1/64	3.29e-05	404.7	0.11	2.00	1.25e-02	79.5	0.33	1.03
1/128	8.36e-06	1797.7	0.27	2.00	6.13e-03	256.6	0.93	1.03
	Finite differences							
1/8	3.43e-02	0.9	0.00					
1/16	1.73e-02	2.3	0.05	0.99				
1/32	8.66e-03	6.9	0.06	1.00				
1/64	4.34e-03	23.1	0.11	1.00				
1/128	2.17e-03	83.0	0.44	1.00				

Table 6.3: Test equation $D^{1/2}y + y = t^2 + \frac{2t^{1.5}}{\Gamma(2.5)}$

h	Predictor corrector (M=2)				Predictor corrector (M=4)			
	error y(1)	KFlops	Time	EOC	error y(1)	KFlops	Time	EOC
1/8	5.99e-03	1.9	0.05		5.53e-04	2.9	0.06	
1/16	1.34e-03	4.0	0.06	2.16	2.11e-04	6.4	0.05	1.39
1/32	3.08e-04	9.6	0.11	2.13	6.12e-05	15.4	0.17	1.79
1/64	7.18e-05	25.2	0.22	2.10	1.63e-05	40.9	0.38	1.91
1/128	1.70e-05	74.9	0.38	2.08	4.20e-06	122.8	0.77	1.96
	Predictor corrector (M=8)				Predictor corrector (M=16)			
1/8	1.06e-03	5.1	0.05		1.06e-03	9.3	0.17	
1/16	2.68e-04	11.2	0.22	1.98	2.68e-04	20.8	0.28	1.98
1/32	6.78e-05	27.0	0.33	1.98	6.78e-05	50.3	0.60	1.99
1/64	1.71e-05	72.4	0.66	1.99	1.71e-05	135.4	1.27	1.99
1/128	4.29e-06	218.5	1.37	1.99	4.29e-06	410.0	2.58	1.99

Table 6.4: Test equation $D^{1/2}y + y = t^2 + \frac{2t^{1.5}}{\Gamma(2.5)}$

h	Predictor corrector (M=1)				Implicit quadrature			
	error y(1)	KFlops	Time	EOC	error y(1)	KFlops	Time	EOC
1/8	1.23e-02	1.3	0.00		3.03e-02	0.9	0.00	
1/16	3.34e-03	2.8	0.06	1.88	1.30e-02	1.9	0.00	1.22
1/32	9.63e-04	6.6	0.05	1.79	5.55e-03	4.5	0.05	1.23
1/64	2.77e-04	17.3	0.17	1.80	2.35e-03	11.9	0.11	1.24
1/128	8.07e-05	51.0	0.27	1.80	9.92e-04	36.1	0.22	1.24
	Approximate Mittag-Leffler				Collocation points 0.1, 0.5, 1.0			
1/8	1.27e-03	6.0	0.00		8.94e-02	5.1	0.05	
1/16	4.02e-04	22.2	0.06	1.66	4.55e-02	11.1	0.11	0.97
1/32	1.16e-04	93.3	0.05	1.79	2.28e-02	27.6	0.22	1.00
1/64	3.20e-005	408.5	0.17	1.86	1.13e-02	79.1	0.44	1.01
1/128	8.62e-006	1799.0	0.28	1.89	5.65e-03	255.8	1.15	1.00
	Finite differences							
1/8	5.43e-02	0.7	0.00					
1/16	2.74e-02	1.8	0.00	0.99				
1/32	1.38e-02	5.8	0.06	0.99				
1/64	6.91e-03	14.7	0.11	1.00				
1/128	3.45e-03	49.9	0.38	1.00				

Table 6.5: Test equation $D^{3/4}y + y = t^2 + \frac{2t^{1.25}}{\Gamma(2.25)}$

h	Predictor corrector (M=2)				Predictor corrector (M=4)			
	error y(1)	KFlops	Time	EOC	error y(1)	KFlops	Time	EOC
1/8	6.33e-04	1.9	0.00		8.21e-04	2.9	0.05	
1/16	2.81e-05	4.0	0.05	4.49	2.16e-04	6.4	0.11	1.93
1/32	1.41e-05	9.6	0.17	0.99	5.58e-05	15.4	0.16	1.95
1/64	7.06e-06	25.2	0.22	1.00	1.43e-05	40.9	0.38	1.96
1/128	2.38e-06	74.9	0.44	1.75	3.64e-06	122.8	0.77	1.97
	Predictor corrector (M=8)				Predictor corrector (M=16)			
1/8	8.46e-04	5.1	0.05		8.46e-04	9.3	0.11	
1/16	2.18e-04	11.2	0.17	1.96	2.18e-04	20.8	0.27	1.96
1/32	5.59e-05	27.0	0.33	1.96	5.59e-05	50.3	0.66	1.96
1/64	1.43e-05	72.4	0.72	1.97	1.43e-05	135.4	1.26	1.97
1/128	3.64e-06	218.5	1.37	1.97	3.64e-06	410.0	2.53	1.97

Table 6.6: Test equation $D^{3/4}y + y = t^2 + \frac{2t^{1.25}}{\Gamma(2.25)}$

h	Predictor corrector (M=1)				Approximate Mittag-Leffler			
	error y(1)	KFlops	Time	EOC	error y(1)	KFlops	Time	EOC
1/8	1.24e-02	1.3	0.00		5.53e-02	6.9	0.06	
1/16	4.71e-03	2.9	0.05	1.40	2.05e-02	25.9	0.11	1.43
1/32	1.73e-03	6.7	0.11	1.45	7.40e-03	106.1	0.06	1.47
1/64	6.25e-04	17.4	0.22	1.47	2.65e-03	455.8	0.17	1.48
1/128	2.24e-04	51.1	0.33	1.48	9.47e-04	1998.2	0.33	1.49
	Collocation points 0.1, 0.5, 1.0				Finite differences			
1/8	2.23e-02	5.1	0.05		1.19e-01	0.7	0.00	
1/16	2.00e-02	11.1	0.11	0.16	6.28e-02	1.7	0.00	0.93
1/32	1.29e-02	27.7	0.22	0.63	3.24e-02	4.7	0.05	0.95
1/64	7.46e-03	79.2	0.49	0.79	1.66e-02	14.5	0.16	0.97
1/128	4.08e-03	255.9	1.38	0.87	8.39e-03	49.5	0.38	0.98

Table 6.7: Test equation $D^{3/2}y + y = t^2 + \frac{2t^{0.5}}{\Gamma(1.5)}$

Case B

In equations of the form 6.1, with $\alpha > 1$, Diethelm and Ford [19] prove that the convergence of the predictor corrector algorithm is of order $O(h^2)$.

We use the two test equations,

1. $D^{3/2}y + y = t^2 + \frac{2t^{0.5}}{\Gamma(1.5)}$, $y(0) = y'(0) = 0$, exact solution $y = t^2$
2. $D^{3/2}y + y = t^3 + \frac{8t^{1.5}}{\Gamma(0.5)}$, $y(0) = y'(0) = 0$, exact solution $y = t^3$

for this section. For equation 1, tables 6.7 and 6.8 show the KFlops, time and error values for a decreasing step size h. We can see that additional corrector iterations do not increase the convergence order of the predictor corrector algorithm. The convergence order of the predictor corrector method and the approximate Mittag-Leffler method, appear to be $O(h^{1.5})$ instead of the expected $O(h^2)$. This is because the fractional derivative is in $C[0, t]$ and thus breaks the algorithm's smoothness assumptions. Proofs and analysis of the smoothness assumptions, for the predictor corrector method and the approximate Mittag-Leffler method can be found in [20] and [24].

For equation 2, in table 6.9 we can see that when a test equation which meets the algorithms assumptions is used, we get the expected convergence.

In both cases the predictor corrector algorithm is the most numerically efficient method.

h	Predictor corrector (M=2)				Predictor corrector (M=4)			
	error y(1)	KFlops	Time	EOC	error y(1)	KFlops	Time	EOC
1/8	1.24e-02	1.9	0.06		1.24e-02	2.9	0.00	
1/16	4.93e-03	4.1	0.05	1.47	4.93e-03	6.6	0.11	1.47
1/32	1.77e-03	9.7	0.17	1.48	1.77e-03	15.7	0.22	1.48
1/64	6.32e-04	25.5	0.27	1.49	6.32e-04	41.6	0.44	1.49
1/128	2.25e-04	75.4	0.55	1.49	2.25e-04	124.1	0.88	1.49
	Predictor corrector (M=8)				Predictor corrector (M=16)			
1/8	1.24e-02	5.2	0.11		1.24e-02	9.7	0.22	
1/16	4.93e-03	11.6	0.22	1.47	4.93e-03	21.6	0.33	1.47
1/32	1.77e-03	27.7	0.38	1.48	1.77e-03	51.8	0.77	1.48
1/64	6.32e-04	73.8	0.77	1.49	6.32e-04	138.3	1.54	1.49
1/128	2.25e-04	221.4	1.65	1.49	2.25e-04	415.9	3.07	1.49

Table 6.8: Test equation $D^{3/2}y + y = t^2 + \frac{2t^{0.5}}{\Gamma(1.5)}$

h	Predictor corrector (M=1)				Approximate Mittag-Leffler			
	error y(1)	KFlops	Time	EOC	error y(1)	KFlops	Time	EOC
1/8	7.37e-03	1.3	0.00		1.99e-02	6.9	0.05	
1/16	1.73e-03	2.9	0.06	2.09	6.19e-03	25.9	0.06	1.69
1/32	4.11e-04	6.7	0.06	2.07	1.74e-03	106.1	0.05	1.83
1/64	9.91e-05	17.4	0.17	2.05	4.68e-04	455.8	0.17	1.90
1/128	2.41e-05	51.1	0.33	2.04	1.23e-04	1998.2	0.33	1.93
	Collocation points 0.1, 0.5, 1.0				Finite differences			
1/8	9.23e-02	5.2	0.06		2.39e-01	0.7	0.00	
1/16	5.39e-02	11.2	0.05	0.77	1.19e-01	1.7	0.05	1.00
1/32	2.84e-02	27.9	0.22	0.92	5.95e-02	4.8	0.11	1.00
1/64	1.45e-02	79.6	0.43	0.97	2.97e-02	14.6	0.11	1.00
1/128	7.31e-03	256.7	1.21	0.99	1.48e-02	49.7	0.33	1.00

Table 6.9: Test equation $D^{3/2}y + y = t^3 + \frac{8t^{1.5}}{\Gamma(0.5)}$

h	Predictor corrector (M=1)			
	error y(1)	KFlops	Time	EOC
1/8	4.81e-02	2.5	0.06	
1/16	1.48e-02	5.3	0.06	1.70
1/32	4.64e-03	11.7	0.16	1.67
1/64	1.50e-03	27.4	0.33	1.63
1/128	4.96e-04	71.2	0.66	1.60

Table 6.10: Test equation $D^{1/2}y = t^3 + \frac{2t^{1.5}}{\Gamma(2.5)} - y^{3/2}$

6.2.2 Non-Linear Equations

In the current literature only one method has been specifically designed for non-linear single-term FDEs, and that is the predictor corrector algorithm. It may be possible to convert some of the other methods to cope with non-linear problems, for example using a Newton-Raphson technique, but as this would be required at each step it could lead to substantially longer run times. Later we will show that the predictor corrector is usually the most numerically efficient method for linear problems, thus as non-linear problems with a Newton-Raphson iteration would possess an extra overhead, we don't explore this further. In this thesis we therefore shall focus only on the predictor corrector method.

The convergence behaviour of two non-linear equations are analysed.

1. $D^{1/2}y = t^3 + \frac{2t^{1.5}}{\Gamma(2.5)} - y^{3/2}$, exact answer $y = t^2$.
2. $D^{1/2}y = t^5 + \frac{2t^{1.5}}{\Gamma(2.5)} - t^2y^{3/2}$, exact answer $y = t^2$.

Tables 6.10 to 6.13 give the convergence behaviour and work required for the non-linear equations 1 and 2. If we compare these results with those from tables 6.3 and 6.4, we can see that the work required for the non-linear equations 1 and 2 is of the same order as for linear equations.

6.3 Graphical Representation of Single-term Fractional Differential Equation Properties

The graphical method introduced in this section condenses and 'linearises' the data without altering the various methods' relative properties.

The 'ideal' numerical method's path will lie close to the bottom left of the graph. This represents a small error for a small computational cost. Where the paths

h	Predictor corrector (M=2)				Predictor corrector (M=4)			
	error y(1)	KFlops	Time	EOC	error y(1)	KFlops	Time	EOC
1/8	1.67e-02	3.9	0.05		2.10e-03	6.7	0.11	
1/16	3.78e-03	8.3	0.16	2.14	9.62e-05	14.1	0.22	4.45
1/32	8.64e-04	18.1	0.22	2.13	2.45e-05	30.8	0.50	1.97
1/64	2.01e-04	42.2	0.55	2.10	1.17e-05	71.9	0.93	1.07
1/128	4.75e-05	109.1	1.09	2.08	3.56e-06	184.8	1.93	1.72
	Predictor corrector (M=8)				Predictor corrector (M=16)			
1/8	9.09e-04	12.4	0.22		9.93e-04	23.6	0.44	
1/16	2.54e-04	25.9	0.44	1.84	2.54e-04	49.5	0.88	1.97
1/32	6.45e-05	56.4	0.88	1.98	6.46e-05	107.6	1.70	1.98
1/64	1.64e-05	131.3	1.76	1.98	1.63e-05	250.1	3.41	1.99
1/128	4.12e-06	336.4	3.57	1.99	4.12e-06	639.5	6.92	1.98

Table 6.11: Test equation $D^{1/2}y = t^3 + \frac{2t^{1.5}}{\Gamma(2.5)} - y^{3/2}$

h	Predictor corrector (M=1)			
	error y(1)	KFlops	Time	EOC
1/8	5.33e-02	2.6	0.05	
1/16	1.70e-02	5.5	0.11	1.65
1/32	5.39e-03	11.9	0.22	1.66
1/64	1.75e-03	28.0	0.33	1.62
1/128	5.80e-04	72.3	0.66	1.59

Table 6.12: Test equation $D^{1/2}y = t^5 + \frac{2t^{1.5}}{\Gamma(2.5)} - t^2y^{3/2}$

h	Predictor corrector (M=2)				Predictor corrector (M=4)			
	error y(1)	KFlops	Time	EOC	error y(1)	KFlops	Time	EOC
1/8	1.94e-02	4.1	0.06		2.23e-03	7.0	0.11	
1/16	4.40e-03	8.5	0.11	2.14	8.59e-05	14.6	0.22	4.70
1/32	1.01e-03	18.5	0.28	2.12	3.41e-05	31.7	0.44	1.33
1/64	2.34e-04	43.2	0.55	2.11	1.50e-05	73.7	0.99	1.18
1/128	5.53e-05	111.0	1.10	2.08	4.50e-06	188.3	1.92	1.74
	Predictor corrector (M=8)				Predictor corrector (M=16)			
1/8	1.14e-03	12.8	0.22		1.23e-03	24.4	0.44	
1/16	3.14e-03	26.7	0.49	1.46	3.16e-03	51.0	0.82	1.36
1/32	8.05e-05	58.1	0.93	5.29	8.06e-05	110.8	1.70	5.29
1/64	2.04e-05	134.6	1.76	1.98	2.04e-05	256.4	3.46	1.98
1/128	5.15e-06	342.9	3.57	1.99	5.15e-06	652.1	6.98	1.99

Table 6.13: Test equation $D^{1/2}y = t^5 + \frac{2t^{1.5}}{\Gamma(2.5)} - t^2 y^{3/2}$

of the methods cross represents a critical value of step size (h), here one method becomes more numerically efficient than the other. The paths of the various methods can be extrapolated to explore whether for smaller step sizes (or equivalently, long run times) any paths cross. The gradient of the path represents the *order of numerical efficiency.* Thus two paths which are parallel will possess the same *order of numerical efficiency* but different levels of absolute error.

The new graphical technique was used to examine the test equation

$$D^{\alpha}y + y = t^2 + \frac{2t^{2-\alpha}}{\Gamma(3-\alpha)}, y(0) = 0, \tag{6.2}$$

exact solution $y = t^2$, for various values of α.

On first glance figure 6.1 (produced from the data in tables 6.1 and 6.2) shows that the implicit quadrature method performs most economically for $\alpha = 1/4$, this would also appear to be the case for all $0 < \alpha < 1/2$. After closer examination however, one can see that the slopes of the implicit quadrature and the predictor corrector ($M = 8$) paths could possibly cross for smaller h. Extrapolating the paths as in figure 6.2 shows that the paths cross when $log_2(error) \approx -18$, which corresponds to an error of 3.81×10^{-6}. This size of error could be achieved using the implicit quadrature method with a step size of $h = 1/404$. In figure 6.3 the step size for the implicit quadrature method has been reduced to $h = 1/1024$ and the predictor corrector $M = 8$ method to $h = 1/256$. The figure shows that the paths of these methods do intersect at the predicted value of $log_2(error) \approx -18$.

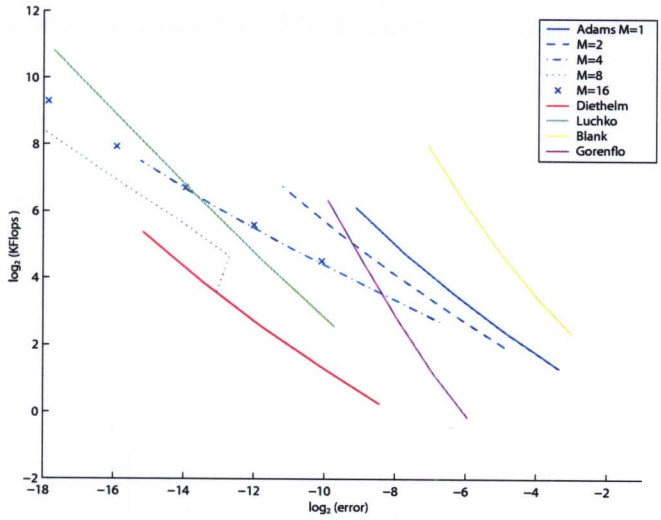

Figure 6.1: $\alpha = 1/4$

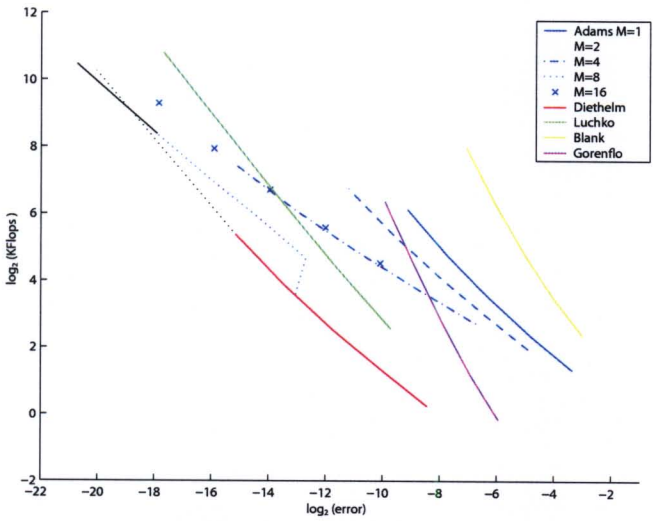

Figure 6.2: $\alpha = 1/4$ extrapolated

Figures 6.4 and 6.5 illustrate that the predictor corrector method with M=2 corrector iterations is the most numerically efficient method when $0.5 \leq \alpha < 1$. When $\alpha > 1$, Figure 6.6 illustrates that the predictor corrector method is still the most numerically efficient, however M=1 corrector iterations is now optimum.

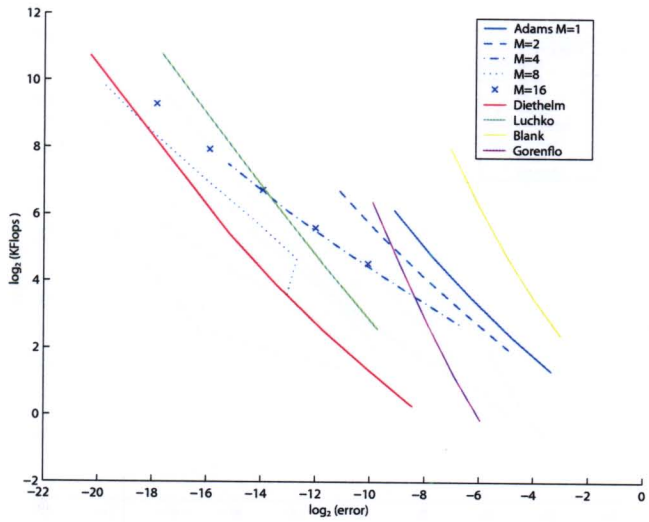

Figure 6.3: $\alpha = 1/4$ extended

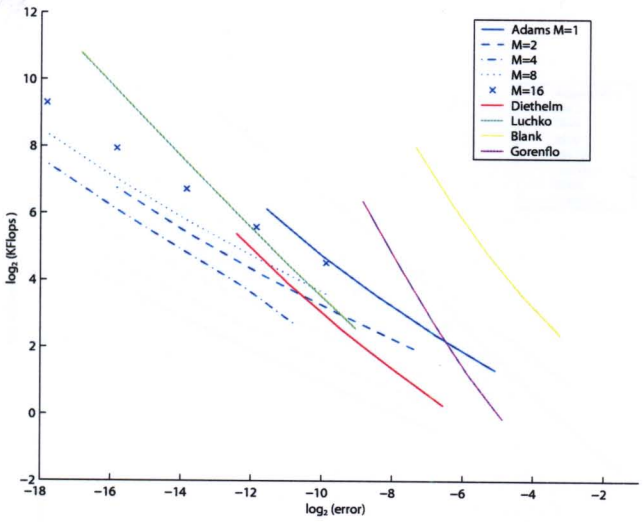

Figure 6.4: $\alpha = 1/2$

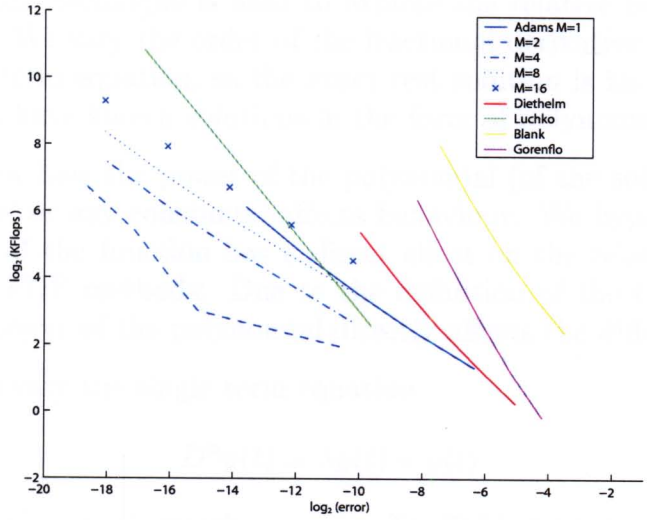

Figure 6.5: $\alpha = 3/4$

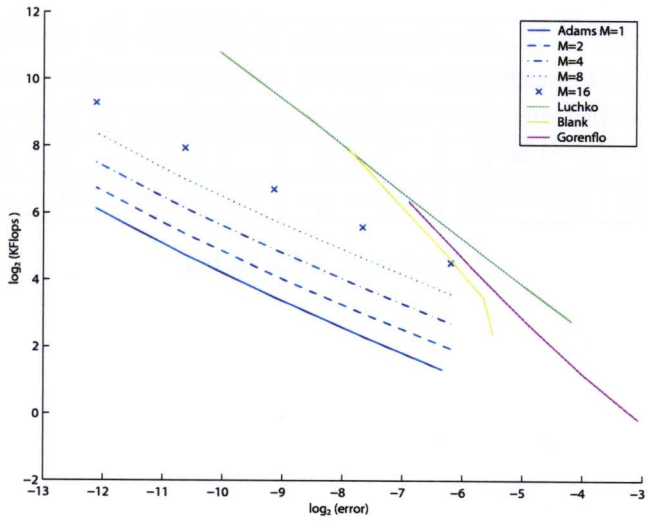

Figure 6.6: $\alpha = 3/2$

6.4 Results

The new graphical technique is used to explore the relative behaviour of the 5 linear methods. We vary the order of the fractional derivative and the function $g(t)$ of a single term equation, so the exact test solution is known. We use test equations which have known solutions in the form of polynomials.

We want to know how the power of the polynomial (of the solution), the order of the derivative, or any constants, affects behaviour. We hypothesize that the differentiability of the function has a direct effect on the relative behaviour of the single-term FDE methods. Due to the definition of the Caputo fractional derivative, the power of the polynomial directly affects the differentiability.

In table 6.14 we vary the single term equation

$$D^{\alpha}y(t) = \lambda y(t) + g(t)$$

so the exact solutions are increasing powers of t. Table 6.14 shows how the power of the exact solution y determines which method is most economical for a fixed value of α. We discover that the differentiability of the function directly affects the relative performance of the methods. This is supported by the experimental data in tables 6.14 to 6.16. It is hypothesized that FDEs with functions which are infinitely differentiable, would continue this behaviour, thus the predictor corrector method would always outperform the implicit quadrature method even with large step size and small α.

	Shows the number of steps (n_D) with $M = 8$ corrections at which the predictor corrector method outperforms the implicit quadrature method for single term equations with the following exact solutions.					
value of α	$y = t^2$	$y = t^3$	$y = t^4$	$y = t^5$	$y = t^7$	$y = t^{10}$
$\alpha = 0.25$	$n_D = 165$	$n_D = 260$	$n_D = 360$	$n_D = 425$	$n_D = 555$	$n_D = 750$

Table 6.14: Power of t

Figure 6.7 illustrates how the power of the exact solution y affects the critical value of step size in a 'stepped' linear manner.

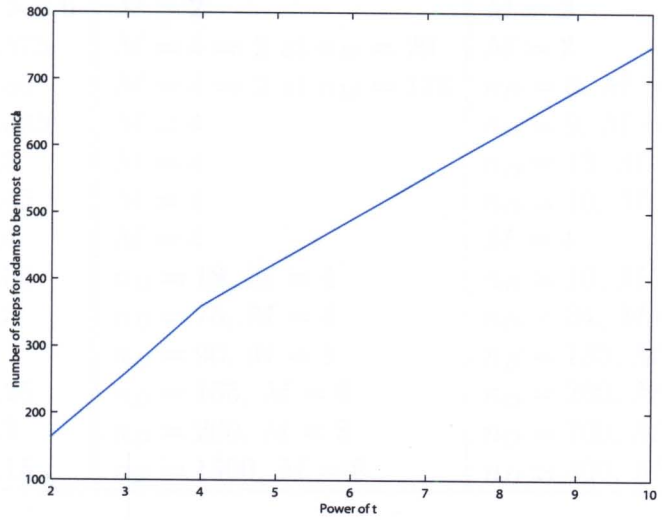

Figure 6.7: (number of steps for predictor corrector method to become the most economical) v (power of t)

Table 6.15 shows how the optimum number of corrector iterations M varies with α.

(n_M) is the number of steps required for a change to occur between the number of optimum corrector iterations.

	Single term equation which has the exact solution.		
value of α	Exact $y = t^2$	Exact $y = t^3$	
$\alpha > 1$	$M = 1$	$M = 1$	
$1 > \alpha > 0.6$	$M = 2$	$M = 2$	
$\alpha = 0.575$	$M = 4 \Rightarrow 2$ at $n_M = 20$	$M = 2$	
$\alpha = 0.55$	$M = 4 \Rightarrow 2$ at $n_M = 128$	$n_D = 6$, $M = 2$	
$\alpha = 0.525$	$M = 4$	$n_D = 9$, $M = 2$	
$\alpha = 0.5$	$M = 4$	$n_D = 13$, $M = 2$	
$\alpha = 0.45$	$M = 4$	$n_D = 10$, $M = 4$	
$\alpha = 0.4$	$M = 4$	$M = 4$	
$\alpha = 0.35$	$n_D = 18$, $M = 4$	$n_D = 10$, $M = 4$	
$\alpha = 0.3$	$n_D = 75$, $M = 4$	$n_D = 34$, $M = 4$	
	$n_D = 90$, $M = 8$	$n_D = 135$, $M = 8$	
$\alpha = 0.25$	$n_D = 165$, $M = 8$	$n_D = 260$, $M = 8$	
$\alpha = 0.2$	$n_D = 260$, $M = 8$	$n_D = 700$, $M = 8$	
$\alpha = 0.15$	$n_D = 1500$, $M = 8$	$n_D \simeq 400$, $M = 8$	

Table 6.15: Varying α

	shows the optimum number of steps (n_D) and corrections (M) at which the predictor corrector method outperforms the implicit quadrature method		
value of α	exact $y = 2t^2$	exact $y = 10t^2$	exact $y = \frac{t^2}{2}$
$\alpha = 0.25$	$n_D = 172$, $M = 8$	$n_D = 172$, $M = 8$	$n_D = 170$, $M = 8$

Table 6.16: Multiple t^2

Table 6.16 shows how varying the coefficient of t^2 has little effect on the optimum number of steps n_D.

The optimum number of steps, n_D, of equations with exact solutions which are compositions of various powers of t (t^2 and t^3 in our chosen case), interpolate between the power's n_D values. In table 6.17 we show how the ratio of the exact solutions constant's determine the % influence of the power connected with each constant. From the table we can see that as the ratio ε/δ increases the % influence increases.

	shows the % influence of the t^2 power
ratio ϵ/δ	$\epsilon t^2 + \delta t^3$
1/1000	0%
1/100	2.3%
1/10	5.5%
1	34.4%
10	83.7%
100	98.4%
1000	100%

Table 6.17: Composite functions influence

6.5 Conclusions

In this chapter we compared the numerical efficiency of several methods for the solution of single-term FDEs of the form 5.1.

The predictor corrector method [19] is recommended in most situations, however the implicit quadrature method [11] performs well for linear problems with $\alpha < 1/2$. In this case, the value of α, the right hand side function $g(t)$ and the number of corrector iterations for the predictor corrector method, determine at which value of step size, if at all, the predictor corrector method outperforms the implicit quadrature method.

The graphical technique introduced in section 6.3 is useful for comparing numerical methods in all mathematical fields. It would seem to be of particular importance for problems that have memory effects. Due to the nature of these algorithms, the work required does not have a linear relationship with time, thus it is possible for the most numerically efficient method to change depending on the length of run time. The graphical technique can be used as a predictive tool, in conjunction with small run times of several competing methods, to determine the most numerically efficient method for any given FDE. In chapter 12 we show the schematics of a black box FDE solver, which uses this graphical technique for calculating solutions to both single and multi-term FDEs. The FDE solver will be available for download from the University of Chester mathematics website at

`www.chester.ac.uk\mathematics`

In the next chapter we discuss multi-term fractional differential equations and introduce an efficient method for their solution.

Chapter 7

Multi-term FDE Methods

7.1 Motivation

Multi-term equations have been introduced into the academic and scientific community with the aim of modelling a variety of physical behaviours. The first example put forward was by Bagley-Torvik [1] in modelling the motion of a rigid plate immersed in a Newtonian fluid. An algorithm tackling the numerical solution of this equation was introduced by Diethelm and Ford [17]. They use the least common multiple of the fractional and whole powers, in a similar manner to that used to solve ordinary differential equations (ODEs) of order > 1, to determine the size of the required system. The underlying principle from this algorithm was generalised in a subsequent paper [18], to enable approximate solutions of all equations of the form (7.1).

In [29] Ford and Simpson introduce an alternative approach, suitable for linear FDEs, in which they minimize the dimension of the system produced, thus reducing computational complexity.

In this chapter we introduce a new algorithm, which again produces a system of equations of low dimension, suitable for both linear and non linear FDEs. In chapter 8 we explore and compare the numerical efficiency of these algorithms and give recommendations of the most suitable algorithm for any particular problem.

We shall be interested in numerical solutions of multi-term FDEs of the general form

$$D^{\alpha_N} y(t) = f(t, y(t), D^{\alpha_1} y(t), \ldots, D^{\alpha_{N-1}} y(t)), \tag{7.1}$$

equipped with initial conditions,

$$y^{(k)}(0) = y_0^{(k)}, k = 0, 1, \ldots, \lceil \alpha_N \rceil - 1, \tag{7.2}$$

assuming $\alpha_N > \alpha_{N-1} > \ldots > \alpha_0$, $\alpha_i - \alpha_{i-1} \leq 1$, $N \in \mathbb{N}$, and $\alpha_i \in \mathbb{Q}$ for all i.

Two of the methods that we shall consider are only applicable in the context of linear equations, which is a subset of (7.1), of the form,

$$\left[D^{\alpha_N} + b_{N-1}D^{\alpha_{N-1}} + \ldots + b_1 D^{\alpha_1} + b_0 D^0\right] y(t) = g(t), \tag{7.3}$$

where $b_i \in \mathbb{R}$, $i = 0, 1, \ldots, N-1$, equipped with initial conditions (7.2).

We shall perform various numerical experiments and comparisons for both linear and non-linear cases.

A linear 'multi-term' fractional differential equation (7.3), where D^{α_i} is the Caputo fractional derivative, can be converted to a system of equations. This can be achieved using three different methods:

1. Diethelm and Ford's method [17], for linear systems (see section (7.2.1)).
2. Ford and Simpson's method [28].
3. New method.

The Ford and Simpson method is not suitable for non-linear problems, for reasons discussed later, so we only compare:

a. Diethelm and Ford's method [17], for non-linear systems (see section (7.2.2)).

b. New method.

7.2 Diethelm and Ford's Method [17]

Diethelm and Ford [17] introduce a numerical scheme for the solution of multi-term FDEs of the form (7.1). We now illustrate how this scheme can be applied to both linear and non-linear FDEs.

7.2.1 Linear Fractional Order Systems

We let $\alpha_N = vq$, where $q \in \mathbb{R}$ is the greatest common divisor of α_i, $i = 0, 1, \ldots, N$.

We can now rewrite the multi-term FDE (7.3) in the form

$$\left[D^{vq} + a_{v-1}D^{(v-1)q} + \ldots + a_1 D^q + a_0 D^0\right] y(t) = g(t), \frac{1}{v}, k \in \mathbb{N}, \tag{7.4}$$

with initial conditions

$$y^{(k)}(0) = y_0^{(k)}, k = 0, 1, \ldots, \lceil vq \rceil - 1.$$

Diethelm and Ford express equation (7.4) as a system of equations of differential order q. The value of q determines the number of equations $1/q$ in the system, and subsequently the amount of processing power required to solve the system. Given the density of the rational numbers and the limiting constraint of finite precision arithmetic of computers, this method can be used to model any fractional differential equation arbitrarily well. The approach may however, lead to very large systems of equations.

Diethelm and Ford [15] introduce theorem 7.2.1 to show that equation 7.4 can be written as a system of equations, with appropriate initial conditions.

Theorem 7.2.1 *The equation*

$$\left[D^{\alpha_N} + b_{N-1}D^{\alpha_{N-1}} + \ldots + b_1 D^{\alpha_1} + b_0 D^0\right] y(t) = g(t), \tag{7.5}$$

equipped with initial conditions

$$y^{(k)}(0) = y_0^{(k)}, k = 0, 1, \ldots, \lceil \alpha_N \rceil - 1,$$

is equivalent to the system of equations

$$\begin{aligned}
D^q\, {}^0Y(t) &= {}^1Y(t), \\
D^q\, {}^1Y(t) &= {}^2Y(t), \\
D^q\, {}^2Y(t) &= {}^3Y(t), \\
&\vdots \\
D^q\, {}^{v-2}Y(t) &= {}^{v-1}Y(t), \\
D^q\, {}^{v-1}Y(t) &= -b_0\, {}^0Y(t) - b_1\, {}^{\alpha_1/q}Y(t) - \ldots - b_{N-1}\, {}^{\alpha_{N-1}/q}Y(t) + g(t)
\end{aligned} \tag{7.6}$$

together with the initial conditions,

$${}^iY(0) = \begin{cases} y_0^{(k)} & \text{if } i = kv \text{ with some } k \in \mathbb{N}, \\ 0 & \text{else,} \end{cases} \tag{7.7}$$

in the following sense.

1. *Whenever* $Y := ({}^0Y, \ldots, {}^{v-1}Y)^T$ *with* ${}^0Y \in C^{\lceil \alpha_N \rceil}[0, c]$ *for some* $c > 0$ *is the solution of the system (7.6), equipped with the corresponding initial conditions, the function* $y := {}^0Y$ *solves the multi-term equation (7.3) and satisfies the initial conditions (7.2).*

2. *Whenever* $y \in C^{\lceil \alpha_N \rceil}[0,c]$ *is a solution of the multi-term equation (7.3) satisfying the initial conditions (7.2), the vector-valued function*
$Y := ({}^0Y, \ldots, {}^vY)^T := (y, D^q y, D^{2q} y, \ldots, D^{(v-1)q} y)^T$
satisfies the system (7.6) and the initial conditions (7.7).

We can now rewrite equation (7.3) in a form similar to that of single-term equations,

$$D^q Y(t) = G(t, Y(t))$$

with initial conditions

$$Y(0) = {}^0Y$$

where $Y = ({}^0Y, {}^1Y, \ldots, {}^{v-1}Y)^T$ is a function of v variables that maps to $\mathbb{R}^v$,

$$G(t, Y(t)) = \begin{pmatrix} {}^1Y(t), \\ {}^2Y(t), \\ {}^3Y(t), \\ \vdots \\ {}^{v-1}Y(t), \\ -b_0\, {}^0Y(t) - b_1\, {}^{\alpha_1/q}Y(t) - \ldots - b_{N-1}\, {}^{\alpha_{N-1}/q}Y(t) + g(t) \end{pmatrix} \tag{7.8}$$

and ${}^0Y = ({}^0Y(0), {}^1Y(0), \ldots, {}^{v-1}Y(0))^T$.

We are now able to apply either the implicit quadrature algorithm from section 5.1 or the predictor corrector algorithm from section 5.2 to solve equation (7.3).

7.2.2 Non-Linear Fractional Order Systems

Diethelm and Ford [15] introduce theorem 7.2.2 to show that equation 7.1 can be written as a system of equations, with appropriate initial conditions.

Theorem 7.2.2 *The equation*

$$D^{\alpha_N} y(t) = f(t, y(t), D^{\alpha_{N-1}} y(t), D^{\alpha_{N-2}} y(t), \ldots, D^{\alpha_1} y(t)), \tag{7.9}$$

equipped with initial conditions,

$$y^{(k)}(0) = y_0^{(k)}, k = 0, 1, \ldots, \lceil \alpha_N \rceil - 1, \tag{7.10}$$

is equivalent to the system of equations

$$
\begin{aligned}
D^q\,{}^0Y(t) &= {}^1Y(t),\\
D^q\,{}^1Y(t) &= {}^2Y(t),\\
D^q\,{}^2Y(t) &= {}^3Y(t),\\
&\vdots\\
D^q\,{}^{v-2}Y(t) &= {}^{v-1}Y(t),\\
D^q\,{}^{v-1}Y(t) &= f(t, Y(t), {}^{\alpha_{N-1}/q}Y(t), \ldots, {}^{\alpha_1/q}Y(t)),
\end{aligned}
\tag{7.11}
$$

together with the initial conditions,

$$
{}^iY(0) = \begin{cases} y_0^{(k)} & \text{if } i = kv \text{ with some } k \in \mathbb{N},\\ 0 & \text{else,} \end{cases}
\tag{7.12}
$$

in the following sense.

1. *Whenever* $Y := ({}^1Y, \ldots, {}^{v-1}Y)^T$ *with* ${}^0Y \in C^{\lceil \alpha_N \rceil}[0, c]$ *for some* $c > 0$ *is the solution of the system (7.11), equipped with the corresponding initial conditions, the function* $y := {}^0Y$ *solves the multi-term equation (7.9) and satisfies the initial conditions (7.10).*

2. *Whenever* $y \in C^{\lceil \alpha_N \rceil}[0, c]$ *is a solution of the multi-term equation (7.9) satisfying the initial conditions (7.10), the vector-valued function* $Y := ({}^0Y, \ldots, {}^{v-1}Y)^T := (y, D^q y, D^{2q} y, \ldots, D^{(v-1)q} y)^T$ *satisfies the system (7.11) and the initial conditions (7.12).*

We can now rewrite equation (7.1) in a form similar to that of single-term equations,

$$D^q Y(t) = F(t, Y(t))$$

with initial conditions

$$Y(0) = {}^0Y$$

where $Y = ({}^0Y, {}^1Y, \ldots, {}^{v-1}Y)^T$ is a function of v variables that maps to $\mathbb{R}^v$,

$$
F(t, Y(t)) = \begin{pmatrix} {}^1Y(t),\\ {}^2Y(t),\\ {}^3Y(t),\\ \vdots\\ {}^{v-2}Y(t),\\ f(t, Y(t), {}^{\alpha_{N-1}/q}Y(t), \ldots, {}^{\alpha_1/q}Y(t)), \end{pmatrix}
\tag{7.13}
$$

and ${}^0Y = ({}^0Y(0), {}^1Y(0), \ldots, {}^{v-1}Y(0))^T$.

We are now able to apply the predictor corrector algorithm from section 5.2 to solve equation (7.1). From this point to distinguish between the single and multi-term methods, we shall refer to the multi-term implicit quadrature as the Diethelm method and the predictor corrector as the Adams method.

7.3 Ford and Simpson's Method [28]

Ford and Simpson produce a method that minimizes the number of equations contained in the system.

For all $\alpha_i > 1$, we perform the mapping $\beta_j = \alpha_i - 1$ for $j = 1, \ldots, N - i$.

Thus for a 5 term test equation of type 7.3, with $\alpha_N = 2$, the equation

$$[D^2 + b_3 D^{\beta_1+1} + b_2 D + b_1 D^{\alpha_1} + b_0]y(t) = g(t), \text{ where } \alpha_1, \beta_1 \in (0,1), b_i \in \mathbb{R} \quad (7.14)$$

subject to initial conditions,

$$y^{(k)}(0) = y_0^{(k)}, k = 0, 1$$

can be written as,

$$\begin{aligned} {}^1Y(t) &= D^{\alpha_1}\, {}^0Y(t) \\ {}^2Y(t) &= D\, {}^0Y(t) \\ {}^3Y(t) &= D^{\beta_1}\, {}^2Y(t) \\ {}^4Y(t) &= D\, {}^2Y(t) \end{aligned} \quad (7.15)$$

together with initial conditions

$${}^kY(0) = \begin{cases} y_0^{(k)} & \text{for } k = 0 \text{ and } k = 2, \\ 0 & \text{else,} \end{cases} \quad (7.16)$$

which in matrix form is

$$\begin{pmatrix} D^{\alpha_1} & 0 & 0 & 0 \\ D & 0 & 0 & 0 \\ 0 & 0 & D^{\beta_1} & 0 \\ 0 & 0 & D & 0 \end{pmatrix} \begin{pmatrix} {}^0Y(t) \\ {}^1Y(t) \\ {}^2Y(t) \\ {}^3Y(t) \end{pmatrix} = \begin{pmatrix} {}^1Y(t) \\ {}^2Y(t) \\ {}^3Y(t) \\ g(t) - \sum_{i=0}^{3} b_i\, {}^iY(t). \end{pmatrix} \quad (7.17)$$

This is discretised using a θ-method, with $\theta = 1/2$ (trapezium rule)

$$Dy = g \Rightarrow y_j = y_{j-1} + (1-\theta)hg_j + \theta h g_{j-1}, \quad (7.18)$$

for the integer value derivatives and the implicit quadrature method (see section 5.1).

This leads to,

$$\begin{pmatrix} -{}^{\alpha_1}w_{0,j} & {}^{\alpha_1}\eta_j & 0 & 0 \\ 1 & 0 & -\frac{h}{2} & 0 \\ 0 & 0 & -{}^{\beta_1}w_{0,j} & {}^{\beta_1}\eta_j \\ \frac{hb_0}{2} & \frac{hb_1}{2} & 1 + \frac{hb_2}{2} & \frac{hb_3}{2} \end{pmatrix} \begin{pmatrix} {}^1Y_j \\ {}^2Y_j \\ {}^3Y_j \\ {}^4Y_j \end{pmatrix} = \begin{pmatrix} {}^1S_j \\ {}^2S_j \\ {}^3S_j \\ {}^4S_j \end{pmatrix}$$

where

$$
\begin{aligned}
{}^1S_j &= \sum_{k=1}^{j} {}^{\alpha_1}w_{k,j}\,{}^1Y_{j-k} + {}^1Y_0/\alpha_1, \\
{}^2S_j &= {}^1Y_{j-1} + \frac{h}{2}{}^3Y_{j-1}, \\
{}^3S_j &= \sum_{k=1}^{j} {}^{\beta_1}w_{k,j}\,{}^3Y_{j-k} + {}^3Y_0/\beta_1, \\
{}^4S_j &= {}^3Y_{j-1} + G_j - \frac{h}{2}\sum_{i=1}^{4} b_{i-1}\,{}^iY_{j-1},
\end{aligned}
$$

$G_j = h(g_j + g_{j-1})/2, g_j = g(jh),$

and ${}^{\alpha_1}\eta_j = (jh)^{\alpha_1}\Gamma(-\alpha_1).$

7.4 Proposed New Method

In section 7.3 we introduced Ford and Simpson's [28] method to solve linear multi-term fractional differential equations. The method uses a similar technique to that of integer order equations, in that it produces a system of equations. The method minimizes the dimension of the required system, thus reducing computational effort. If we look at equation (7.17), the left-hand side matrix appears singular. However, the D^{α}s are a notational device, and when the system is discretized, as the Ford and Simpson method uses an explicit quadrature approach, the apparent problem is solved. However, if a predictor corrector discretization was applied, the left-hand side matrix would become singular. Therefore the Ford and Simpson method is only suitable for linear FDEs. The only method currently available for the solution of non-linear multi-term FDEs is Diethelm and Ford's method [18]. This method, however, often produces large systems of equations, resulting in prohibitory computational effort.

The new approach developed here produces a system of equations of an implicit nature thus enabling a predictor corrector algorithm to be used. As in the Ford and Simpson method, the aim is to minimize the size of the system. Due to the convergence order of the $P(EC)^ME$ being $O(h^2)$, as $M \to \infty$, our new method generally has better convergence than the Ford and Simpson method. We show that the optimum number of corrector iterations can be determined from the orders of the Caputo derviatives. This also allows us to demonstrate that for

the vast majority of cases the new method outperforms the Ford and Simpson method, for linear problems, on the grounds of numerical efficiency.

For all $\alpha_i > 1$, we perform the mapping $\beta_j = \alpha_i - 1$ for $j = 1, \ldots, N - i$.

We now rewrite the non-linear multi-term FDE (7.1) to include the above mapping.

7.4.1 Five Term Equation

For the equation,

$$D^2 y = f(t, y(t), D^{\alpha_1} y(t), Dy(t), D^{\beta_1+1} y(t)), \text{ where } \alpha_1, \beta_1 \in (0, 1) \tag{7.19}$$

subject to initial conditions,

$$y^{(k)}(0) = y_0^{(k)}, k = 0, 1$$

we have the following equivalence theorem.

Theorem 7.4.1 *The equation (7.19), with the relevant initial conditions, is equivalent to the system of equations*

$$\begin{aligned} {}^1Y(t) &= D^{\alpha_1} y(t) = D^{\alpha_1}\ {}^0Y(t) \\ {}^2Y(t) &= Dy(t) = D^{1-\alpha_1}\ {}^1Y(t) \\ {}^3Y(t) &= D^{\beta_1+1} y(t) = D^{\beta_1}\ {}^2Y(t) \\ {}^4Y(t) &= D^2 y(t) = D^{2-\beta_1}\ {}^3Y(t) \end{aligned} \tag{7.20}$$

together with initial conditions

$${}^kY(0) = \begin{cases} y_0^{(k)} & \text{for } k = 0 \text{ and } k = 2, \\ 0 & \text{else.} \end{cases} \tag{7.21}$$

Expressed in matrix form gives

$$\begin{pmatrix} D^{\alpha_1} & 0 & 0 & 0 \\ 0 & D^{1-\alpha_1} & 0 & 0 \\ 0 & 0 & D^{\beta_1} & 0 \\ 0 & 0 & 0 & D^{2-\beta_1} \end{pmatrix} \begin{pmatrix} {}^0Y(t) \\ {}^1Y(t) \\ {}^2Y(t) \\ {}^3Y(t) \end{pmatrix} = \begin{pmatrix} {}^1Y(t) \\ {}^2Y(t) \\ {}^3Y(t) \\ f(t,\ {}^0Y(t),\ {}^1Y(t),\ {}^2Y(t),\ {}^3Y(t)) \end{pmatrix}.$$

Proof. The proof follows as a direct result of the commutable property of the Caputo derivative [56]. i.e.

$$D^{\alpha} D^{\beta} y(t) = D^{\beta} D^{\alpha} y(t) = D^{\alpha+\beta} y(t), \qquad \forall \qquad \alpha, \beta \in \mathbb{N}_+.$$

A predictor-corrector algorithm can now be applied as a difference algorithm.

Example 1

Our first example considers a non-linear version of the Bagley-Torvik equation with an exact solution of $y = t^3$.

The equation

$$D^2y + 4tD^{1.5}y + 5y^2 = 6t + \frac{32t^{2.5}}{\Gamma(.5)} + 5t^6, \tag{7.22}$$

subject to initial conditions,

$$y(0) = y'(0) = 0,$$

can be written as

$$\begin{pmatrix} D^{1/2} & 0 & 0 & 0 \\ 0 & D^{1/2} & 0 & 0 \\ 0 & 0 & D^{1/2} & 0 \\ 0 & 0 & 0 & D^{1/2} \end{pmatrix} \begin{pmatrix} {}^0Y \\ {}^1Y \\ {}^2Y \\ {}^3Y \end{pmatrix} = \begin{pmatrix} {}^1S \\ {}^2S \\ {}^3S \\ {}^4S \end{pmatrix},$$

where

$$\begin{pmatrix} {}^1S \\ {}^2S \\ {}^3S \\ {}^4S \end{pmatrix} = \begin{pmatrix} {}^1Y \\ {}^2Y \\ {}^3Y \\ 6t + \frac{32t^{2.5}}{\Gamma(.5)} + 5t^6 - 5({}^0Y)^2 - 4t\,{}^3Y \end{pmatrix}.$$

Note that even though there is no derivative between 0 and 1, we include a $D^{1/2}$ derivative, but with zero coefficient.

Example 2

The next example demonstrates how we can decrease the size of the system required (compared with the Adams method from section 7.2.2), thus reducing the numerical complexity.

The equation

$$D^2y + t^2D^{1.8}y + D^{0.32}y + 5y^2 = 6t + \frac{25t^{3.2}}{\Gamma(.2)} + \frac{1.959739120t^{2.68}}{\Gamma(.68)} + 5t^6, \tag{7.23}$$

subject to initial conditions,

$$y(0) = y'(0) = 0,$$

can be written as

$$\begin{pmatrix} D^{0.32} & 0 & 0 & 0 \\ 0 & D^{0.68} & 0 & 0 \\ 0 & 0 & D^{0.8} & 0 \\ 0 & 0 & 0 & D^{0.2} \end{pmatrix} \begin{pmatrix} {}^0Y \\ {}^1Y \\ {}^2Y \\ {}^3Y \end{pmatrix} = \begin{pmatrix} {}^1S \\ {}^2S \\ {}^3S \\ {}^4S \end{pmatrix},$$

where

$$\begin{pmatrix} {}^1S \\ {}^2S \\ {}^3S \\ {}^4S \end{pmatrix} = \begin{pmatrix} {}^1Y \\ {}^2Y \\ {}^3Y \\ 6t + \frac{25t^{3.2}}{\Gamma(.2)} + \frac{1.959739120t^{2.68}}{\Gamma(.68)} + 5t^6 - 5({}^0Y)^2 - ({}^1Y) - t^2\ ({}^3Y) \end{pmatrix}.$$

7.4.2 General Multi-term Equations

More generally, for the equation,

$$D^2y = f(t, y(t), D^{\alpha_1}y(t), \ldots, D^{\alpha_m}y(t), Dy(t), D^{1+\beta_1}y(t), \ldots, D^{1+\beta_{N-m}}y(t)), \tag{7.24}$$

where $\alpha, \beta \in (0, 1)$.

subject to initial conditions,

$$y^{(k)}(0) = y_0^{(k)}, k = 0, 1$$

we give the following new equivalence theorem.

Theorem 7.4.2 *The equation (7.24), with the relevant initial conditions, is equivalent to the system of equations*

$$\begin{array}{lcll} {}^1Y(t) & = & y(t), & \\ {}^2Y(t) & = & D^{\alpha_1}y(t) & = D^{\alpha_1}\ {}^1Y(t) \\ {}^3Y(t) & = & D^{\alpha_2}y(t) & = D^{\alpha_2-\alpha_1}\ {}^2Y(t) \\ \vdots & & & \\ {}^{m+1}Y(t) & = & D^{\alpha_m}y(t) & = D^{\alpha_m-\alpha_{m-1}}\ {}^mY(t) \\ {}^{m+2}Y(t) & = & Dy(t) & = D^{1-\alpha_m}\ {}^{m+1}Y(t) \\ {}^{m+3}Y(t) & = & D^{\beta_1+1}y(t) & = D^{\beta}\ {}^{m+2}Y(t) \\ \vdots & & & \\ {}^{N+1}Y(t) & = & D^{\beta_{N-m}}y(t) & = D^{\beta_{N-m}-\beta_{N-m-1}}\ {}^NY(t) \\ {}^{N+2}Y(t) & = & D^2y(t) & = D^{1-\beta_{N-m}}\ {}^{N+1}Y(t) \end{array} \tag{7.25}$$

together with initial conditions

$$ {}^{k}Y(0) = \begin{cases} y_0^{(k)} & \text{for } k = 0 \text{ and } k = m+1, \\ 0 & \text{else,} \end{cases} \tag{7.26} $$

Expressed in matrix form gives

$$ \begin{pmatrix} D^{\alpha_1} & & & & & & \\ & D^{\alpha_2-\alpha_1} & & & 0 & & \\ & & \ddots & & & & \\ & & & D^{1-\alpha_m} & & & \\ & & & & D^{\beta_1} & & \\ & & & & & D^{\beta_2-\beta_1} & \\ & & & & & & \ddots \\ & & 0 & & & & D^{2-\beta_{N-m}} \end{pmatrix} \begin{pmatrix} {}^{1}Y(t) \\ {}^{2}Y(t) \\ \vdots \\ {}^{N+2}Y(t) \end{pmatrix} = \begin{pmatrix} {}^{1}S \\ {}^{2}S \\ \vdots \\ {}^{N+2}S \end{pmatrix} \tag{7.27} $$

where

$$ \begin{pmatrix} {}^{1}S \\ {}^{2}S \\ \vdots \\ {}^{N+2}S \end{pmatrix} = \begin{pmatrix} {}^{2}Y(t) \\ {}^{3}Y(t) \\ \vdots \\ f(t,\, {}^{0}Y(t),\, {}^{1}Y(t), \ldots,\, {}^{N+1}Y(t)) \end{pmatrix}. $$

Proof. The proof follows as a direct result of the commutable property of the Caputo derivative [56]. i.e.

$$ D^{\alpha}D^{\beta}y(t) = D^{\beta}D^{\alpha}y(t) = D^{\alpha+\beta}y(t), \qquad \forall \qquad \alpha, \beta \in \mathbb{N}_{+}. $$

Example

Our example considers a non-linear equation with multiple β powers, exact solution of $y = t^3$.

The equation

$$ D^2y + t^2D^{1.8}y + 4tD^{1.5}y + 5y^2 = 6t + \frac{25t^{3.2}}{\Gamma(.2)} + \frac{32t^{2.5}}{\Gamma(.5)} + 5t^6, \tag{7.28} $$

subject to initial conditions,

$$ y(0) = y'(0) = 0, $$

can be written as

$$\begin{pmatrix} D^{1/2} & 0 & 0 & 0 & 0 \\ 0 & D^{1/2} & 0 & 0 & 0 \\ 0 & 0 & D^{1/2} & 0 & 0 \\ 0 & 0 & 0 & D^{0.3} & 0 \\ 0 & 0 & 0 & 0 & D^{0.2} \end{pmatrix} \begin{pmatrix} {}^0Y \\ {}^1Y \\ {}^2Y \\ {}^3Y \\ {}^4Y \end{pmatrix} = \begin{pmatrix} {}^1S \\ {}^2S \\ {}^3S \\ {}^4S \\ {}^5S \end{pmatrix},$$

where

$$\begin{pmatrix} {}^1S \\ {}^2S \\ {}^3S \\ {}^4S \\ {}^5S \end{pmatrix} = \begin{pmatrix} {}^1Y \\ {}^2Y \\ {}^3Y \\ {}^4Y \\ 6t + \frac{25t^{3.2}}{\Gamma(.2)} + \frac{32t^{2.5}}{\Gamma(.5)} + 5t^6 - 5({}^0Y)^2 - 4t\,({}^3Y) - t^2\,({}^4Y) \end{pmatrix}.$$

Note that even though there is no derviative between 0 and 1, we include a $D^{1/2}$ derivative, but with zero coefficient.

7.4.3 Convergence

We now prove that the new method converges to the desired result as $h \to 0$.

Firstly we rename the orders of the left diagonal matrix of equation (7.27), i.e. $\alpha_1, \alpha_2 - \alpha_1, \ldots, 2 - \beta_{n-m} = \Psi_j \qquad j = 1, n.$

In lemma 7.4.3 we give our new convergence result.

Lemma 7.4.3 *Assuming that the solution y of the initial value problem (7.24) is such that:*

PREDICTOR

$$\left| \int_0^{t_{k+1}} (t_{k+1} - x)^{\alpha-1} D^\alpha y(x) dx - \sum_{j=0}^{k} b_{j,k+1} D^\alpha y(t_j) \right| \leq C_1 t_{k+1}^{\gamma_1} h^{\delta_1} \tag{7.29}$$

CORRECTOR

$$\left| \int_0^{t_{k+1}} (t_{k+1} - x)^{\alpha-1} D^\alpha y(x) dx - \sum_{j=0}^{k+1} a_{j,k+1} D^\alpha y(t_j) \right| \leq S_1 t_{k+1}^{\gamma_2} h^{\delta_2} \tag{7.30}$$

with some $\gamma_1, \gamma_2 \geq 0$ and $\delta_1, \delta_2 > 0$. Then, for some suitably chosen $T > 0$, we have

$$\max_{0 \leq j \leq N} |y(t_j) - y_j| = O(h^q) \tag{7.31}$$

where $q = \min\{\delta_1 + \alpha, \delta_2\}$ and $N = \lfloor T/h \rfloor$

Proof. Our proof is based upon the proof on lemma 3.1 of [20]. For h sufficiently small, we show that,

$$|y(t_j) - y_j| \leq Ch^q \tag{7.32}$$

$\forall\ j \in \{0, 1, \ldots, n\}$, where C is a constant of suitable size. Mathematical induction will be the basis of the proof. As the initial conditions are given, the induction basis ($j = 0$) is accepted. As in [20], (7.32) is now assumed true for $j = 0, 1, 2, \ldots, k$, for some $k \leq n - 1$. We now prove that the inequality holds for $j = k + 1$.

We first concentrate on the behaviour of the predictor.

Recall that, after the application of the first order Ψ_1, we have,

$$\begin{aligned}
|y(t_{k+1}) - y^P_{k+1}| &= \frac{1}{\Gamma(\Psi_1)} \left| \int_0^{t_{k+1}} (t_{k+1} - t)^{\Psi_1 - 1} f(t, y(t)) dt - \sum_{j=0}^{k} b_{j,k+1} f(t_j, y_j) \right| \\
&\leq \frac{1}{\Gamma(\Psi_1)} \left| \int_0^{t_{k+1}} (t_{k+1} - t)^{\Psi_1 - 1} D^{\Psi_1} y(t) dt - \sum_{j=0}^{k} b_{j,k+1} D^{\Psi_1} y(t_j) \right| \\
&\quad + \frac{1}{\Gamma(\Psi_1)} \sum_{j=0}^{k} b_{j,k+1} |f(t_j, y(t_j)) - f(t_j, y_j)| \\
&\leq \frac{C_1 t_{k+1}^{\gamma_1}}{\Gamma(\Psi_1)} h^{\delta_1} + \frac{1}{\Gamma(\Psi_1)} \sum_{j=0}^{k} b_{j,k+1} LCh^q \\
&\leq \frac{C_1 T^{\gamma_1}}{\Gamma(\Psi_1)} h^{\delta_1} + \frac{CLT^{\Psi_1}}{\Gamma(\Psi_1 + 1)} h^q.
\end{aligned} \tag{7.33}$$

We then repeat for Ψ_2, thus,

$$\begin{aligned}
|y(t_{k+1}) - y^P_{k+1}| &= \frac{1}{\Gamma(\Psi_2)} \left| \int_0^{t_{k+1}} (t_{k+1} - t)^{\Psi_2 - 1} f(t, y(t)) dt - \sum_{j=0}^{k} b_{j,k+1} f(t_j, y_j) \right| \\
&\leq \frac{1}{\Gamma(\Psi_2)} \left| \int_0^{t_{k+1}} (t_{k+1} - t)^{\Psi_2 - 1} D^{\Psi_2} y(t) dt - \sum_{j=0}^{k} b_{j,k+1} D^{\Psi_2} y(t_j) \right| \\
&\quad + \frac{1}{\Gamma(\Psi_2)} \sum_{j=0}^{k} b_{j,k+1} |f(t_j, y(t_j)) - f(t_j, y_j)|.
\end{aligned}$$

Now substituting as before, but now including the first predictor error bound (7.33), we obtain the new error bound,

$$|y(t_{k+1}) - y^P_{k+1}| \leq \frac{C_2 t_{k+1}^{\gamma_2}}{\Gamma(\Psi_2)} h^{\delta_1} + \frac{1}{\Gamma(\Psi_2)} \sum_{j=0}^{k} b_{j,k+1} L \left(\frac{C_1 T^{\gamma_1}}{\Gamma(\Psi_1)} h^{\delta_1} + \frac{CLT^{\Psi_1}}{\Gamma(\Psi_1 + 1)} h^q \right)$$

$$
\begin{aligned}
&\leq \frac{C_2 T^{\gamma_2}}{\Gamma(\Psi_2)} h^{\delta_1} + \frac{L T^{\Psi_2}}{\Gamma(\Psi_2+1)} \left(\frac{C_1 T^{\gamma_1}}{\Gamma(\Psi_1)} h^{\delta_1} + \frac{CLT^{\Psi_1}}{\Gamma(\Psi_1+1)} h^q \right) \\
&= \frac{C_2 T^{\gamma_2}}{\Gamma(\Psi_2)} h^{\delta_1} + \frac{C_1 L T^{\gamma_1} T^{\Psi_2}}{\Gamma(\Psi_2+1)\Gamma(\Psi_1)} h^{\delta_1} + \frac{CL^2 T^{\Psi_1} T^{\Psi_2}}{\Gamma(\Psi_2+1)\Gamma(\Psi_1+1)} h^q.
\end{aligned}
$$

This process is continued sequentially for all Ψs in the system, thus obtaining an error bound for the predictor of,

$$
\begin{aligned}
|y(t_{k+1}) - y^P_{k+1}| &= \left(\frac{C_n T^{\gamma_n}}{\Gamma(\Psi_n)} + \frac{C_{n-1} L T^{\gamma_{n-1}} T^{\Psi_n}}{\Gamma(\Psi_n+1)\Gamma(\Psi_{n-1})} \right. \\
&\quad \left. + \ldots + \frac{C_1 L^{n-1} T^{\gamma_1} T^{\Psi_n} \ldots T^{\Psi_2}}{\Gamma(\Psi_n+1)\ldots\Gamma(\Psi_2+1)\Gamma(\Psi_1)} \right) h^{\delta_1} \\
&\quad + \left(\frac{CL^n T^{\Psi_n} \ldots T^{\Psi_1}}{\Gamma(\Psi_n+1)\ldots\Gamma(\Psi_1+1)} \right) h^q. \qquad (7.34)
\end{aligned}
$$

By choosing T sufficiently small, we can now bound the terms of the first parentheses by C_{p_1} and the terms of the second parentheses by $C_{p_2}C$, thus for the predictor

$$
|y(t_{k+1}) - y^P_{k+1}| \leq C_{p_1} h^{\delta_1} + C_{p_2} C h^q. \qquad (7.35)
$$

We now use the error bound (7.34) in the analysis of the corrector.

Similar to above we have,

$$
\begin{aligned}
|y(t_{k+1}) - y_{k+1}| &= \frac{1}{\Gamma(\Psi_1)} \left| \int_0^{t_{k+1}} (t_{k+1} - t)^{\Psi_1 - 1} f(t, y(t)) dt - \sum_{j=0}^{k+1} a_{j,k+1} f(t_j, y_j) \right| \\
&\leq \frac{1}{\Gamma(\Psi_1)} \left| \int_0^{t_{k+1}} (t_{k+1} - t)^{\Psi_1 - 1} D^{\Psi_1} y(t) dt - \sum_{j=0}^{k+1} a_{j,k+1} D^{\Psi_1} y(t_j) \right| \\
&\quad + \frac{1}{\Gamma(\Psi_1)} \sum_{j=0}^{k} a_{j,k+1} |f(t_j, y(t_j)) - f(t_j, y_j)| \\
&\quad + \frac{1}{\Gamma(\Psi_1)} a_{k+1,k+1} |f(t_{k+1}, y(t_{k+1})) - f(t_{k+1}, y^P_{k+1})| \qquad (7.36) \\
&\leq \frac{S_1 t_{k+1}^{\zeta_1}}{\Gamma(\Psi_1)} h^{\delta_2} + \frac{SL}{\Gamma(\Psi_1)} h^q \sum_{j=0}^{k} a_{j,k+1} \\
&\quad + a_{k+1,k+1} \frac{L}{\Gamma(\Psi_1)} \left(C_{p_1} h^{\delta_1} + C_{p_2} h^q \right) \\
&\leq \left(\frac{S_1 T^{\zeta_1}}{\Gamma(\Psi_1)} + \frac{SLT^{\Psi_1}}{\Gamma(\Psi_1+1)} + \frac{C_{p_1} L}{\Gamma(\Psi_1+2)} + \frac{C_{p_2} CL}{\Gamma(\Psi_1+2)} h^{\Psi_1} \right) h^q.
\end{aligned}
$$

We repeat for Ψ_2 and substitute (7.36) giving,

$$\begin{aligned}
|y(t_{k+1}) - y_{k+1}| \leq & \frac{S_2 t_{k+1}^{\zeta_2}}{\Gamma(\Psi_2)} h^{\delta_2} + \frac{SL}{\Gamma(\Psi_2)} h^q \sum_{j=0}^{k} a_{j,k+1} + a_{k+1,k+1} \frac{L}{\Gamma(\Psi_2)} \left(\frac{S_1 T^{\zeta_1}}{\Gamma(\Psi_1)} \right. \\
& \left. + \frac{SLT^{\Psi_1}}{\Gamma(\Psi_1+1)} + \frac{C_{p_1} L}{\Gamma(\Psi_1+2)} + \frac{C_{p_2} CL}{\Gamma(\Psi_1+2)} h^{\Psi_1} \right) h^q \\
\leq & \left(\frac{S_2 T^{\zeta_2}}{\Gamma(\Psi_2)} + \frac{SLT^{\Psi_2}}{\Gamma(\Psi_2+1)} + \frac{S_1 LT^{\zeta_1}}{\Gamma(\Psi_1)\Gamma(\Psi_2+2)} \right. \\
& + \frac{SL^2 T^{\Psi_1}}{\Gamma(\Psi_1+1)\Gamma(\Psi_2+2)} + \frac{C_{p_1} L^2}{\Gamma(\Psi_1+2)\Gamma(\Psi_2+2)} \\
& \left. + \frac{C_{p_2} CL^2}{\Gamma(\Psi_1+2)\Gamma(\Psi_2+2)} h^{\Psi_2} \right) h^q.
\end{aligned}$$

This process is continued sequentially for all Ψs in the system, then by choosing a sufficiently small T and a sufficiently large C, we can bound each summand in the parentheses and thus obtain an entire upper error bound of Ch^q.

7.5 Conclusions

In this chapter we have introduced the reader to the concept of multi-term FDEs. We described algorithms by Diethelm and Ford [17] and Ford and Simpson [28]. We explained how the Ford and Simpson method is only suitable for linear problems, whereas the Diethelm and Ford method is suitable for both linear and non-linear problems

We then introduced a new method, specifically designed to optimize numerical efficiency for non-linear equations, but which also is very efficient for most linear equations. In the next chapter we now explore and compare the numerical efficiency of these algorithms and give recommendations of the most suitable algorithm for any particular problem.

Chapter 8

Numerical Comparison of Multi-term Methods

In this chapter we compare the numerical efficiency of the multi-term methods introduced in Chapter 7. We first determine the optimum number of corrector iterations for the Adams method (see Section 7.2.2) and the new method (see Section 7.4). We then use the graphical technique introduced in Section 6.3 to determine the most numerically efficient method for various classes of Multi-term FDE.

8.1 Optimum Number of Corrector Iterations

In section 6.4, we showed how the optimum number of corrector iterations for Diethelm and Ford's predictor corrector method [19], varies depending on the order of the fractional derivative and the function $g(t)$.

We now continue this theme for multi-term FDEs.

We use the test equations,

$$D^2y + D^{\alpha_{N-1}}y + \ldots + D^{\alpha_1}y + y = g(t), \tag{8.1}$$

where $0 < \alpha_i < 2$, for $i = 1, N-1$, and

$$Dy + D^{\alpha_{N-1}}y + \ldots + D^{\alpha_1}y + y = g(t), \tag{8.2}$$

where $0 < \alpha_i < 1$, for $i = 1, N-1$, with the exact solution $y = t^3$, to determine the optimum number of corrector iterations for both the new method introduced in (7.4) and Diethelm's method described in (7.2.2).

In section 6.3 we introduced a graphical technique which compares FDE methods for numerical efficiency.

We use this technique to discover the optimum number of corrector iterations for various multi-term FDEs of forms (8.1) and (8.2).

In tables 8.1 and 8.3, we use the test equation

$$D^{\alpha_N} y + D^{\alpha_1} y + y = g(t),$$

with exact solution $y = t^3$, for a series of $0 < \alpha < \alpha_N$, for $\alpha_N = 1, 2$ respectively.

In section 7.2.2, we showed that a multi-term FDE can be rewritten as a system of equations of order q, where q is the greatest common divisor of the derivatives. In table 8.1, the notation $4 \rightarrow 6$ at $err = 2.2e-05$, for the $\alpha = 1.8$ row, means that the optimum number of corrector iterations changes from 4 to 6 when the step size decreases to such a level that the error drops below $2.2e-05$. We can see in tables 8.1 and 8.3 that the q value of any particular multi-term equation has a direct relation to the number of corrector iterations required for optimal efficiency of the Adams method. For example, in both the cases where $q = 2$ the optimum number of corrector iterations is 2. Whereas the optimum number of corrector iterations for the new method is due to the difference between the value of α_1 and the nearest integer order derivative.

In tables 8.2 and 8.4, we use the test equation

$$D^{\alpha_N} y + D^{\alpha_{N-1}} y + \ldots + D^{\alpha_1} y + y = g(t),$$

with exact solution $y = t^3$, for a series of $0 < \alpha_i < \alpha_N$, $i = 1, N-1$, for $\alpha_N = 1, 2$ respectively. Again the q value has a direct relation to the number of corrector iterations required for optimal efficiency of the Adams method. The optimum number of corrections for the new method is again due to the difference between the values of α_i and the nearest derivative (including all integer order derivatives). The α_i value which requires the highest number of corrector iterations from tables 8.1 and 8.3 is dominant.

	Optimum number of corrector iterations		
value of α	q	new method	Adams
$\alpha = 1.95$	0.05	22	$24 \rightarrow 23$ at $err = 3.1e-05$
$\alpha = 1.9$	0.1	10	$12 \rightarrow 11$ at $err = 3.3e-05$
$\alpha = 1.8$	0.2	$4 \rightarrow 6$ at $err = 2.2e-05$	6
$\alpha = 1.7$	0.1	$5 \rightarrow 4$ at $err = 0.0084$	$12 \rightarrow 11$ at $err = 8.36e-05$
$\alpha = 1.6$	0.2	2	6
$\alpha = 1.5$	0.5	2	2
$\alpha = 1.4$	0.2	$3 \rightarrow 2$ at $err = 0.012$	$6 \rightarrow 5$ at $err = 1.2e-05$
$\alpha = 1.3$	0.1	3	$12 \rightarrow 11$ at $err = 0.00032$
$\alpha = 1.2$	0.2	3	$6 \rightarrow 5$ at $err = 5.3e-05$
$\alpha = 1.1$	0.1	3	$12 \rightarrow 11$ at $err = 0.00056$
$\alpha = 1.05$	0.05	3	$24 \rightarrow 23$ at $err = 0.00049$
$\alpha = 0.99$	0.01	$6 \rightarrow 5$ at $err = 0.0014$	126
$\alpha = 0.95$	0.05	5	$23 \rightarrow 22$ at $err = 0.00079$
$\alpha = 0.9$	0.1	$3 \rightarrow 5$ at $err = 0.0072$	$12 \rightarrow 11$ at $err = 5.5e-05$
$\alpha = 0.8$	0.2	$2 \rightarrow 4$ at $err = 0.0018$	$6 \rightarrow 5$ at $err = 0.00028$
$\alpha = 0.7$	0.1	2	$12 \rightarrow 11$ at $err = 0.00087$
$\alpha = 0.6$	0.2	2	$6 \rightarrow 5$ at $err = 0.00035$
$\alpha = 0.5$	0.5	2	2
$\alpha = 0.45$	0.05	2	23
$\alpha = 0.4$	0.2	2	6
$\alpha = 0.35$	0.05	2	23
$\alpha = 0.3$	0.1	2	12
$\alpha = 0.25$	0.05	2	$5 \rightarrow 4$ at $err = 0.0014$
$\alpha = 0.2$	0.2	2	$6 \rightarrow 5$ at $err = 0.0002$
$\alpha = 0.15$	0.05	$2 \rightarrow 3$ at $err = 2.4e-05$	$24 \rightarrow 23$ at $err = 0.00056$
$\alpha = 0.10$	0.1	$2 \rightarrow 3$ at $err = 7.5e-05$	$12 \rightarrow 11$ at $err = 0.0007$
$\alpha = 0.05$	0.05	$2 \rightarrow 3$ at $err = 0.0002$	$23 \rightarrow 22$ at $err = 4.8e-05$
$\alpha = 0.01$	0.01	$2 \rightarrow 3$ at $err = 0.0003$	126

Table 8.1: Multi-term equation with highest derivative 2 (single fractional derivative)

	Optimum number of corrector iterations	
value of α	new method	Adams
$\alpha = 0.5\&0.95$	3	$23 \rightarrow 22$ at $err = 0.00037$
$\alpha = 1.5\&1.9$	$8 \rightarrow 6$ at $err = 2.2e-05$	$12 \rightarrow 11$ at $err = 7.5e-05$
$\alpha = 1.5\&1.6$	$2 \rightarrow 4$ at $err = 0.0025$	$12 \rightarrow 11$ at $err = 0.0002$
$\alpha = 1.5\&1.51$	$2 \rightarrow 4$ at $err = 0.00017$	126
$\alpha = 0.95\&1.5\&1.6$	$6 \rightarrow 3$ at $err = 0.00017$	$24 \rightarrow 23$ at $err = 0.00039$
$\alpha = 0.25\&0.5\&1.95$	$22 \rightarrow 20$ at $err = 1.4e-05$	$24 \rightarrow 23$ at $err = 0.00029$

Table 8.2: Multi term equation with highest derivative 2 (multiple fractional derivatives)

	Optimum number of corrector iterations	
value of α	new method	Adams
$\alpha = 0.95$	$24 \rightarrow 22$ at $err = 7.1e-08$	$18 \rightarrow 19$ at $err = 2.2e-05$
$\alpha = 0.9$	12	$9 \rightarrow 8$ at $err = 6.1e-05$
$\alpha = 0.8$	6	$4 \rightarrow 6$ at $err = 2.3e-06$
$\alpha = 0.75$	4	$5 \rightarrow 3$ at $err = 0.0015$
$\alpha = 0.7$	4	$9 \rightarrow 8$ at $err = 0.00085$
$\alpha = 0.6$	$4 \rightarrow 3$ at $err = 2.4e-05$	$4 \rightarrow 7$ at $err = 0.00024$
$\alpha = 0.5$	3	3
$\alpha = 0.4$	3	$8 \rightarrow 7$ at $err = 6.2e-06$
$\alpha = 0.3$	$4 \rightarrow 3$ at $err = 0.00026$	$8 \rightarrow 7$ at $err = 0.00058$
$\alpha = 0.25$	$4 \rightarrow 3$ at $err = 2.3e-06$	$7 \rightarrow 6$ at $err = 3.2e-05$
$\alpha = 0.2$	4	$8 \rightarrow 3$ at $err = 8.2e-05$
$\alpha = 0.10$	4	$7 \rightarrow 6$ at $err = 6.5e-05$
$\alpha = 0.05$	4	$14 \rightarrow 13$ at $err = 9.6e-05$
$\alpha = 0.02$	4	$36 \rightarrow 35$ at $err = 0.00011$
$\alpha = 0.01$	4	$74 \rightarrow 72$ at $err = 0.00035$

Table 8.3: Multi term equation with highest derivative 1 (single fractional derivative)

	Optimum number of corrector iterations	
value of α	new method	Adams
$\alpha = 0.3\&0.5$	5	$7 \rightarrow 6$ at $err = 7.6e-06$
$\alpha = 0.15\&0.5$	5	$14 \rightarrow 13$ at $err = 6.1e-05$
$\alpha = 0.4\&0.8$	$2 \rightarrow 4$ at $err = 0.0003$	$8 \rightarrow 6$ at $err = 0.00028$
$\alpha = 0.5\&0.9$	$8 \rightarrow 10$ at $err = 1.1e-06$	$9 \rightarrow 8$ at $err = 0.00034$

Table 8.4: Multi term equation with highest derivative 1 (multiple fractional derivatives)

8.2 Stability of the Adams Method and the New Method

The Diethelm and Ford method [17] of converting a multi-term equation into a system of equations (see Section 7.2.2) and the new method (see section 7.4), used in conjunction with the predictor corrector algorithm, can be used to solve both linear and non-linear multi-term equations. When constants contained in such equations are greater than a particular value, instability occurs.

Consider the linear multi-term equation,

$$D^2y + a_1D^\alpha y + a_0y = g(t) \tag{8.3}$$

with an appropriate function $g(t)$ depending upon α, to give an exact solution $y = t^3$.

When $a_1 > C$, where C is a constant which depends upon the corresponding fractional derivative and the step size, the algorithm becomes unstable.

In this section we concentrate on the Adams method, however from experimentation we have discovered that the new method would produce similar results.

For a fixed step size h, as α increases the value C required for instability decreases. If we plot α v C, we shall call the line this makes (for a certain step size) the constant stability threshold. Table 8.5 and the corresponding figure 8.1 show how, for a fixed step size ($h = 1/8$), the constant stability threshold varies.

Of more importance however, is knowledge of what step size is required to overcome the stability problem for a particular equation. For a series of α values, we gradually increase the number of steps until we reach a value where stability occurs. Table 8.6 illustrates how the number of steps required for stability varies for equation (8.3) depending upon the values a_1 and α.

We can see that for α close to 2 the number of steps required for stability grows exponentially with increasing a_1. This soon makes the computational cost of the algorithm prohibitive.

Constant stability threshold	
α	C
1.9	1.30
1.8	1.69
1.7	2.14
1.6	2.78
1.5	3.79
1.4	4.65
1.3	5.87
1.2	7.78
1.1	9.74

Table 8.5: Determining the constant stability threshold

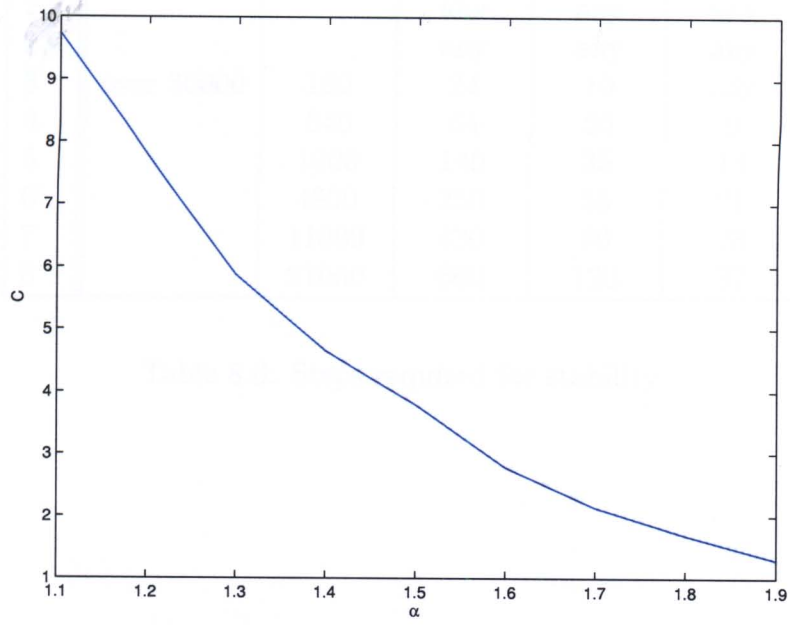

Figure 8.1: Constant stability threshold

	min steps required (2 sig figs)				
a_1	$\alpha = 1.9$	$\alpha = 1.8$	$\alpha = 1.7$	$\alpha = 1.6$	$\alpha = 1.5$
1	any	any	any	any	any
2.0	660	19	any	any	any
2.1	1140		any	any	any
2.2	1900		any	any	any
2.3	3100		any	any	any
2.4	4700		any	any	any
2.5	7100		any	any	any
2.6	11000		any	any	any
2.7	16000		any	any	any
2.8			any	any	any
2.9			any	any	any
3	over 30000	160	24	10	any
4		640	64	20	9
5		1900	140	35	14
6		4800	250	55	21
7		11000	420	80	28
8		21000	660	120	37

Table 8.6: Steps required for stability

The stability problem is due to the algorithm breaking a fundamental definition of the fractional derivative. The fractional derivative is defined to monotonically interpolate between neighbouring integer order derivatives. Take the system (8.4) as an example.

$$\begin{pmatrix} D_{1/2} & 0 & 0 & 0 \\ 0 & D_{1/2} & 0 & 0 \\ 0 & 0 & D_{1/2} & 0 \\ 0 & 0 & 0 & D_{1/2} \end{pmatrix} \begin{pmatrix} {}^1Y \\ {}^2Y \\ {}^3Y \\ {}^4Y \end{pmatrix} = \begin{pmatrix} {}^2Y \\ {}^3Y \\ {}^4Y \\ g(t) - {}^4Ya_1 - {}^1Ya_0 \end{pmatrix}. \quad (8.4)$$

When the step size is too large, the value $g(t) - {}^4Ya_1 - {}^1Ya_0$ possesses the wrong sign, this leads to the fractional derivative not interpolating monotonically between the integer orders, and instability results.

To test for stability at a certain step size, we thus need to determine whether the fractional derivative interpolates monotonically. This can be done by checking whether the value ${}^iY$ is monotonic at each step. In practice, if the results start monotonic they continue to be. Therefore the test only has to be executed once.

8.3 Convergence Behaviour (Multi-term Equations)

In this section we investigate the numerical behaviour of the algorithms introduced in sections 7.2 to 7.4.

Convergence Order

Each of the four methods can possess a different order of convergence.

The order of convergence is described as,

$$\max_{0 \le j \le n} |y(t_j) - y_j| = O(h^p).$$

Diethelm	$p = 2 - q_d$
Adams	$p = 1 + q_a$, or 2 with sufficient correctors
new method	$p = 1 + q_c$, or 2 with sufficient correctors
Ford and Simpson	$p = 1 + q_s$

These orders can be determined from the specific FDE equation involved, but in slightly different ways. For example consider the test equation,

$$D^2y + D^{1.4}y + y = g(t). \quad (8.5)$$

The Diethelm method's 'q_d' value is determined from the order of the system (7.6) which is the greatest common denominator of the orders in equation (8.5), therefore $q_d = 0.2$. The Adams method's 'q_a' value is determined from the order of the system (7.11). Which is the greatest common denominator of the orders in equation (8.5), therefore $q_a = 0.2$.

The new method's 'q_c' value is determined from the smallest difference of subsequent derivatives. Thus the order of convergence of the new method without corrector iterations is $O(h^{1+\min(\alpha_i-\alpha_{i-1})})$, for all α_i, $i = 0, \ldots, N$. All integer order derivatives should be included in the calculation, thus for the test equation (8.5) $q_c = 0.4$.

For the Ford and Simpson method the convergence order is given by $O(h^{1+\min(\lceil\alpha_i\rceil-\alpha_i)})$, for all $\alpha_i \notin \mathbb{N}$, $i = 1, \ldots, N$. Thus, for the example equation (8.5) $q_s = 0.6$.

8.4 How to Compare Multi-term Methods Relative Efficiency

We use the graphical method developed in section 6.3 to find the optimum number of corrector iterations for the newly introduced method, and then to compare the numerical efficiency of the four competing methods. As an example, consider the equation

$$D^2y + D^\alpha y + y = g(t), \qquad y(0) = y'(0) = 0, \tag{8.6}$$

possessing an exact solution of $y = t^3$, using two different α values, $\alpha = 1.25$ and $\alpha = 1.75$. We vary the right side function $g(t)$ accordingly.

There are two different representations of the graphical technique:

a. the error variation with time.

b. the error variation with KFlops.

In figures 8.2 and 8.3 both of these are illustrated for the two α values.

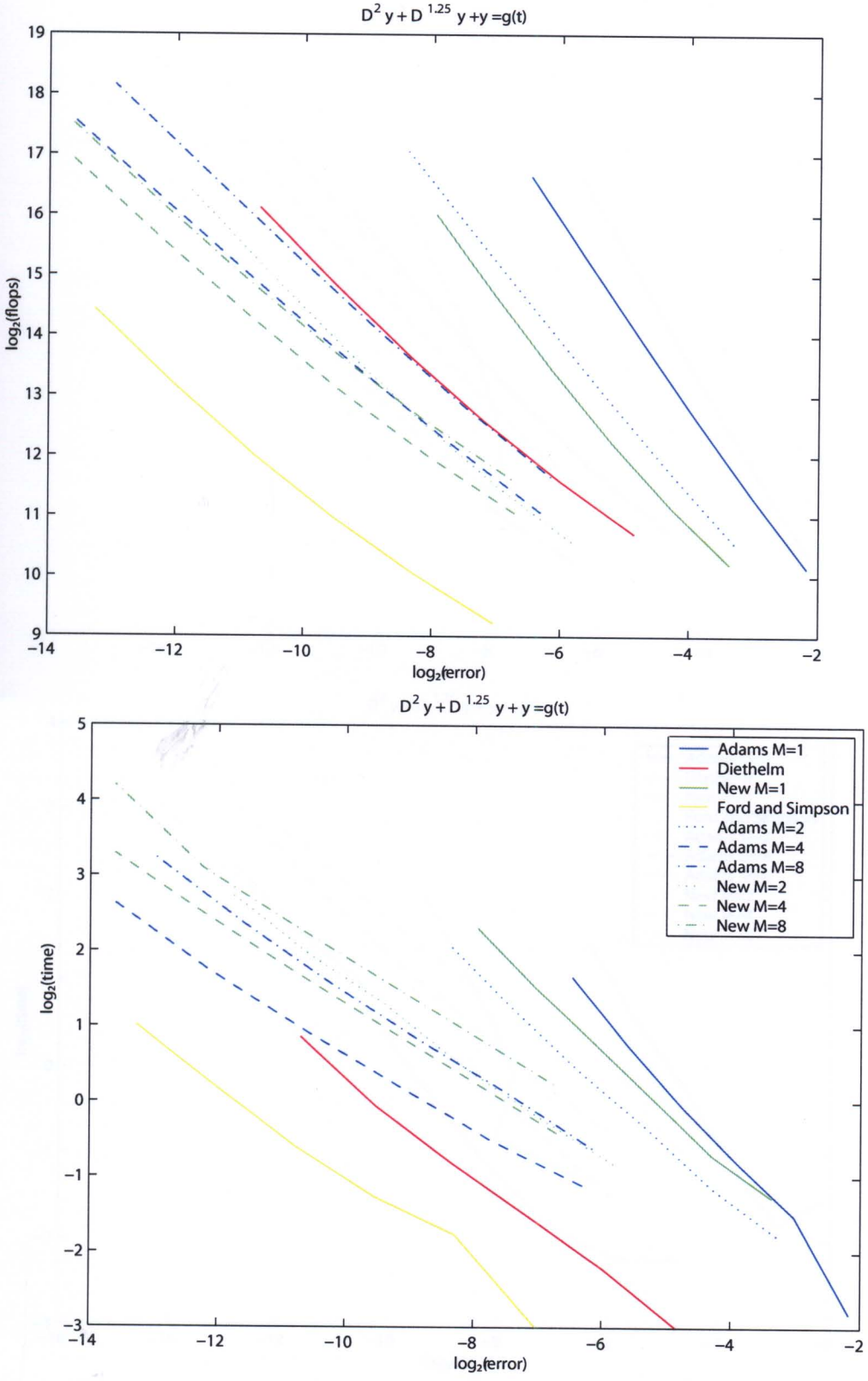

Figure 8.2: $\log_2$(error) v $\log_2$(KFlops)/$\log_2$(time) comparison

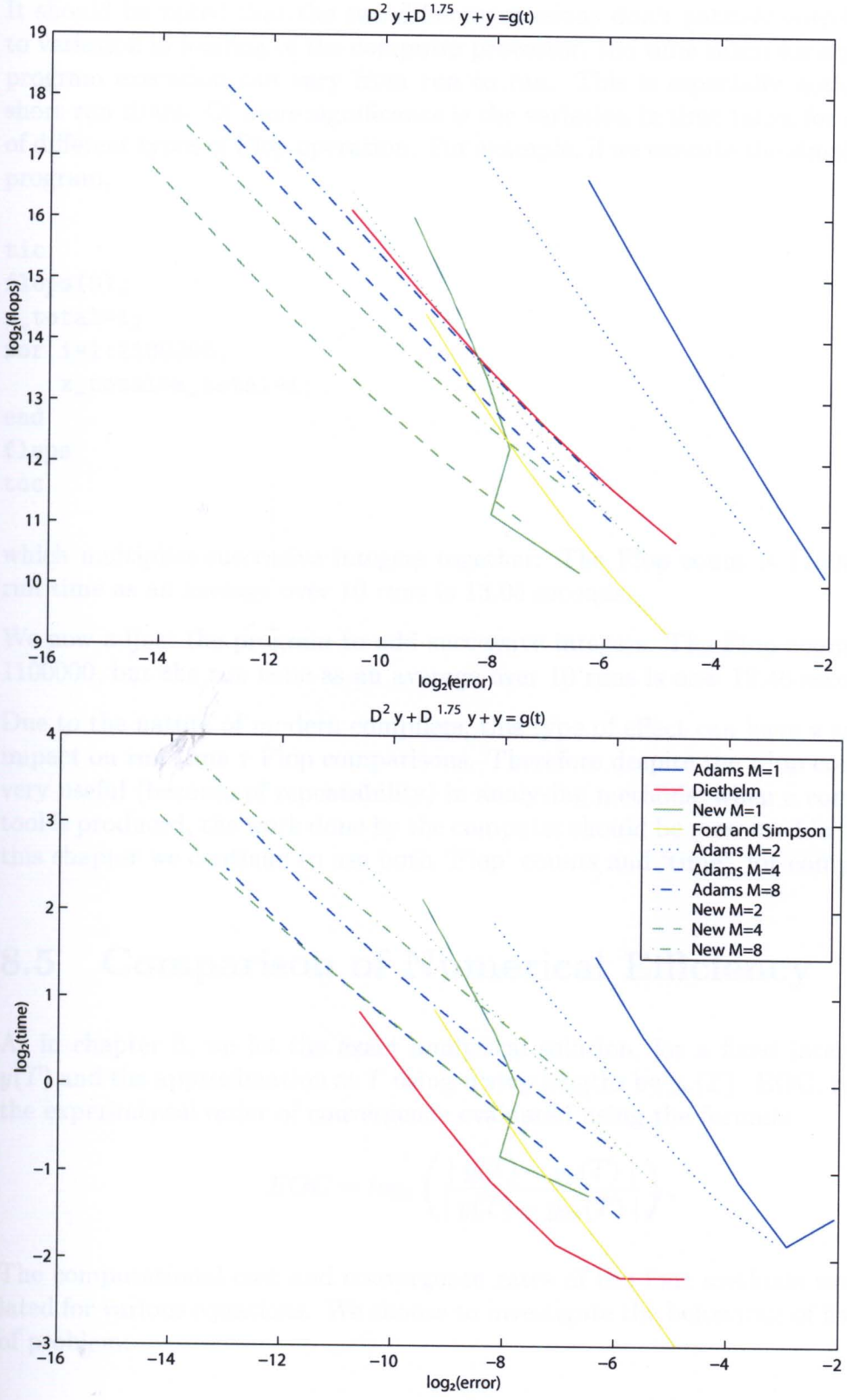

Figure 8.3: $\log_2$(error) v $\log_2$(KFlops)/$\log_2$(time) comparison

It should be noted that the two different versions don't entirely correlate. Due to variation in loading of the computer processor, the time taken for a particular program execution can vary from run to run. This is especially noticeable for short run times. Of more significance is the variation in time taken for execution of different types of Flop operation. For example, if we execute the simple Matlab program,

```
tic
flops(0);
x_total=1;
for i=1:1100000,
    x_total=x_total*i;
end
flops
toc
```

which multiplies successive integers together. The Flop count is 1100000. The run time as an average over 10 runs is 13.05 seconds.

We now adjust the program to add successive integers. The Flop count is again 1100000, but the run time as an average over 10 runs is now 12.46 seconds.

Due to the nature of modern compilers, this type of effect can have a significant impact on run time v Flop comparisons. Therefore despite the Flop count being very useful (because of repeatability) in analysing methods, when a comparative tool is produced, the work done by the computer should be measured in 'time'. In this chapter we continue to use both 'Flop' counts and 'time', for completeness.

8.5 Comparison of Numerical Efficiency

As in chapter 5, we let the exact analytical solution, for a fixed interval T be $y(T)$ and the approximation at T using n step lengths be $y_n(T)$. EOC, represents the experimental order of convergence evaluated using the formula

$$EOC = log_2 \left(\frac{\mid y(T) - y_n(T) \mid}{\mid y(T) - y_{2n}(T) \mid} \right).$$

The computational cost and convergence rates of the four methods were calculated for various equations. We choose to investigate the behaviour of five classes of problem:

1. $Dy + b_1 D^{\alpha} y + b_0 y = g(t)$, where $b_1, b_0 \in \mathbb{R}, 0 < \alpha < 1$.

2. $D^2y + b_2Dy + b_1D^\alpha y + b_0y = g(t)$, where $b_2, b_1, b_0 \in \mathbb{R}, 0 < \alpha < 1$.

3. $D^2y + b_2D^\alpha y + b_1Dy + b_0y = g(t)$, where $b_1, b_0, a_0 \in \mathbb{R}, 1 < \alpha < 2$.

4. $D^2y + b_{N-1}D^{\alpha_{N-1}}y + \ldots + b_1D^{\alpha_1}y + b_0y = g(t)$, where $b_i \in \mathbb{R}, 0 < \alpha_i < 2$, for $i = 1, \ldots, N-1$.

5. $D^{\alpha_N}y(t) = f(t, y(t), D^{\alpha_{N-1}}y(t), \ldots, D^{\alpha_1}y(t))$, where $0 < \alpha_i \leq 2$ for $i = 1, \ldots, N$.

The test problems are chosen so that we obtain solutions in the form of polynomials, primarily $y = t^3$. We use this so the solution and Caputo derivative are in $C^2[0, T]$. We show some examples where the solution is $y = t^2$, and highlight how the convergence order can be affected. In all the examples we look at error approximations over the interval $[0, 1]$, and give the approximate errors at $t = 1$.

8.5.1 Obtaining $g(t)$ for a Particular Solution

If a solution of $y = t^2$ is required, for a Bagley-Torvik equation,

$$D^2y(t) + D^\alpha y(t) + y(t) = g(t),$$

we get

$$D^2y(t) = 2,$$

for the second derivative. The fractional component is obtained using Caputo's definition (2.4.1).

Therefore for example if $\alpha = 3/2$

$$D^{3/2}y(t) = \frac{1}{\Gamma(1/2)} \int_0^t \frac{2}{(t-\tau)^{1/2}} d\tau$$

integrating by parts gives,

$$D^{3/2}y(t) = \frac{4t^{1/2}}{\Gamma(1/2)}$$

therefore,

$$g(t) = t^2 + 2 + \frac{4t^{1/2}}{\Gamma(1/2)}.$$

8.5.2 Case 1: $Dy(t) + b_1 D^{\alpha} y(t) + b_0 y(t) = g(t)$

There are two test equations considered in this section:

(a) $Dy + D^{1/2}y + y = t^2 + 2t + \frac{2.66667t^{1.5}}{\Gamma(0.5)}$, where $y(0) = 0, y'(0) = 0$.

(b) $Dy + D^{3/4}y + y = t^2 + 2t + \frac{6.4t^{1.25}}{\Gamma(0.25)}$, where $y(0) = 0, y'(0) = 0$.

Both of these equations have an exact solution of $y = t^2$. The solution at $y = 1$, is calculated for various step sizes.

Table 8.7 shows the error, KFlops and time at $y = 1$, for the equation $Dy + D^{1/2}y + y = t^2 + 2t + \frac{2.66667t^{1.5}}{\Gamma(0.5)}$. We plot these results using the new graphical technique in figure 8.4. The Adams method with 3 corrector iterations appears the most economical both in terms of KFlops and time. For relatively large step sizes, the paths of both the Adams method and the new method with 3 corrector iterations, are quite erratic. If the sign of the error switches, monotonic decay of the error fails. This is often the case for these methods when using the optimum number of corrector iterations in conjunction with relatively large step sizes. In the limit as $h \to 0$ the paths stabilize. The reader should be aware that the Adams method and the new method, for this equation, implement exactly the same algorithm, and the difference in performance is due to programming issues. Thus some columns of error data are the same, for the Adams and new methods.

In Table 8.8 and the corresponding figure 8.5 we consider the equation, $Dy + D^{3/4}y + y = t^2 + 2t + \frac{6.4t^{1.25}}{\Gamma(0.25)}$. If we use flops as our measure of efficiency, the new method with $m = 6$ is the most economical. If we use time as our measure of efficiency, the Adams method with $m = 5$ is the most economical, however as $h \to 0$ it is unclear which method is best.

The above behaviour is, as expected, due to the nature of the systems produced for each type of method. For equation (a) both the Adams and the new method produce the same system of size 2, producing the same error values, but because of implementation differences the new method has higher numerical cost. For equation (b) the new method still produces a system size 2, however the Adams method produces a system of size 4. Thus the Adams computational cost increases substantially.

Note that both the Adams method ($M = 3$) and the new method ($M = 3$) have a negative EOC from $h = 1/32 \to 1/64$. This is because, when $M > 1$ the error requires small step sizes to propagate through the algorithm.

h	Adams Method				Diethelm and Ford			
	error y(1)	KFlops	Time	EOC	error y(1)	KFlops	Time	EOC
1/32	7.70e-04	9.5	0.00		2.84e-03	7.6	0.00	
1/64	1.96e-04	27.2	0.05	1.97	1.02e-03	20.3	0.06	1.48
1/128	5.18e-05	87.1	0.06	1.92	3.66e-04	61.1	0.05	1.48
1/256	1.41e-05	305.2	0.22	1.88	1.31e-04	204.0	0.11	1.48
1/512	3.78e-06	1134.7	0.44	1.90	4.67e-05	735.6	0.33	1.49
	Ford and Simpson				new method			
1/32	9.90e-04	7.0	0.00		7.70e-04	18.7	0.05	
1/64	3.57e-04	16.9	0.00	1.47	1.96e-04	49.5	0.06	1.97
1/128	1.28e-04	46.1	0.11	1.48	5.18e-05	147.9	0.17	1.92
1/256	4.60e-05	141.3	0.22	1.48	1.41e-05	492.2	0.39	1.88
1/512	1.66e-05	479.1	0.49	1.47	3.78e-06	1770.7	0.88	1.90
	Adams optimum (M=3)				new method, optimum (M=3)			
1/32	2.85e-06	18.4	0.05		2.85e-06	36.0	0.06	
1/64	8.23e-06	53.2	0.05	-1.53	8.23e-06	96.4	0.17	-1.53
1/128	3.59e-06	171.8	0.16	1.20	3.59e-06	291.0	0.33	1.20
1/256	1.41e-06	605.8	0.44	1.35	1.41e-06	975.0	0.82	1.35
1/512	6.88e-07	2260.0	0.94	1.04	6.88e-07	3522.7	1.92	1.04

Table 8.7: $Dy + D^{1/2}y + y = t^2 + 2t + \frac{2.66667t^{1.5}}{\Gamma(0.5)}$

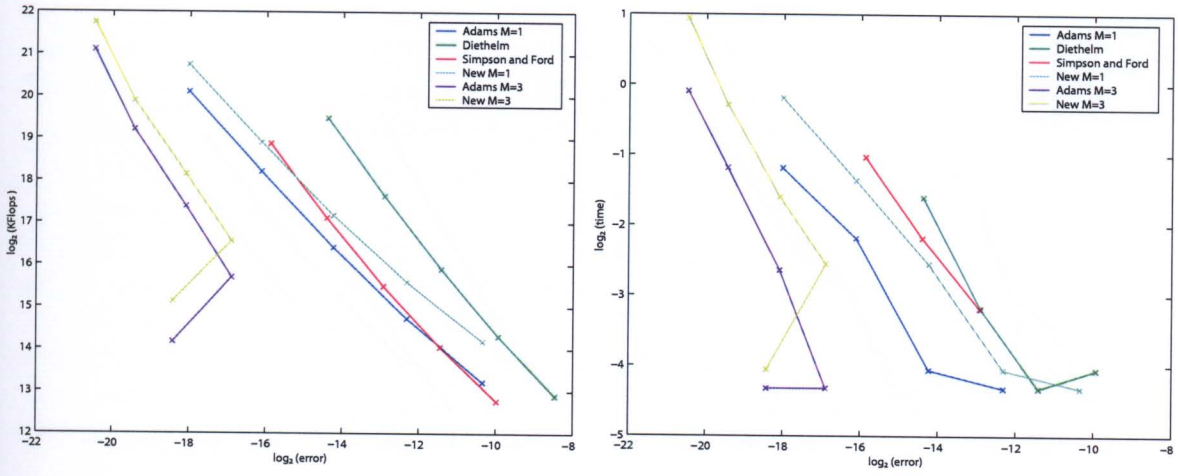

Figure 8.4: $Dy + D^{1/2}y + y = t^2 + 2t + \frac{2.66667t^{1.5}}{\Gamma(0.5)}$

h	Adams method				Diethelm and Ford			
	error y(1)	KFlops	Time	EOC	error y(1)	KFlops	Time	EOC
1/32	2.31e-02	14.8	0.00		1.33e-03	16.0	0.05	
1/64	9.47e-03	45.9	0.05	1.29	4.14e-04	41.2	0.06	1.68
1/128	3.91e-03	157.3	0.11	1.28	1.28e-04	119.1	0.05	1.69
1/256	1.63e-03	576.6	0.16	1.26	3.92e-05	385.5	0.17	1.71
1/512	6.76e-04	2201.7	0.50	1.27	1.19e-05	1360.7	0.33	1.72
	Ford and Simpson				new method			
1/32	3.64e-03	7.0	0.00		1.14e-02	18.7	0.06	
1/64	1.54e-03	16.9	0.05	1.24	4.72e-03	49.5	0.06	1.27
1/128	6.52e-04	46.1	0.11	1.24	1.92e-03	147.9	0.11	1.30
1/256	2.75e-04	141.3	0.22	1.25	7.77e-04	492.2	0.38	1.31
1/512	1.16e-04	479.1	0.49	1.25	3.15e-04	1770.7	0.88	1.30
	Adams optimum (M=5)				new method, optimum (M=6)			
1/32	1.09e-04	43.8	0.05		5.39e-05	62.0	0.17	
1/64	3.74e-05	136.8	0.16	1.54	1.42e-05	166.9	0.28	1.92
1/128	1.13e-05	470.1	0.27	1.73	3.95e-06	505.6	0.61	1.85
1/256	3.20e-06	1726.5	0.66	1.82	1.10e-06	1699.2	1.37	1.84
1/512	8.71e-07	6598.7	1.54	1.88	2.98e-07	6150.8	3.29	1.88

Table 8.8: $Dy + D^{3/4}y + y = t^2 + 2t + \frac{6.4t^{1.25}}{\Gamma(0.25)}$

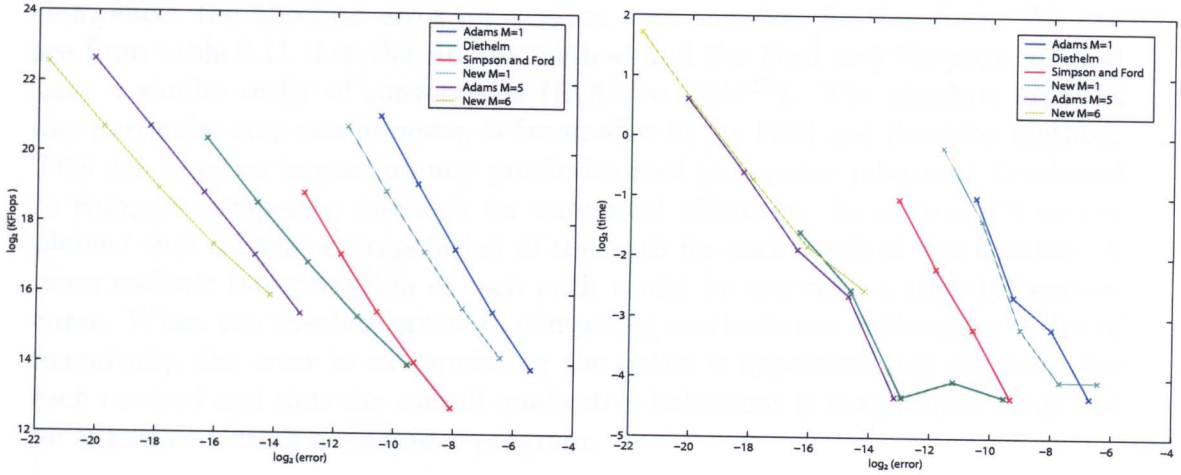

Figure 8.5: $Dy + D^{3/4}y + y = t^2 + 2t + \frac{6.4t^{1.25}}{\Gamma(0.25)}$

8.5.3 Case 2: $D^2y(t) + b_0Dy(t) + a_1D^\alpha y(t) + a_0y(t) = g(t)$

There are three test equations for this section:

(a) $D^2y + D^{1/2}y + y = t^3 + 6t + \frac{3.2t^{2.5}}{\Gamma(0.5)}$, where $y(0) = 0, y'(0) = 0$.

(b) $D^2y + D^{1/2}y + y = t^2 + 2 + \frac{2.6666666667t^{1.5}}{\Gamma(0.5)}$, where $y(0) = 0, y'(0) = 0$.

(c) $D^2y + D^{3/4}y + y = t^3 + 6t + \frac{8.533333333t^{2.25}}{\Gamma(0.25)}$, where $y(0) = 0, y'(0) = 0$.

Equations (a) and (c) have exact solutions $y = t^3$, equation (b) has an exact solution $y = t^2$.

For equations (a) and (b) we show how the differentiability of $g(t)$ effects the convergence behaviour of multi-term FDE methods, and thus can alter which method is most numerically efficient. Note that $D^{1/2}(t^3) \in C^2[0,T]$, and $D^{1/2}(t^2) \in C[0,T]$. For equation (a) in table 8.9 and figure 8.6, we can see that the Adams method (M=2) is the optimum method, however for equation (b) in table 8.10 and figure 8.7, the Ford and Simpson method retains its convergence order whereas both the Adams method and the new method's convergence order are reduced, thus the Ford and Simpson method becomes optimum.

Figure 8.8 from equation (c) demonstrates how the $log_2(KFlops)$ and $log_2(time)$ plots, can produce vastly different results. From the 'KFlops' plot, the new method appears to be the most efficient, however in the 'time' plot, the Adams method is better in terms of numerical efficiency.

Figure 8.8 also shows how constants associated with the methods and the equation, effect the absolute error for a given step size (see Section 6.1). We can see from table 8.11 that the Adams method and the Ford and Simpson method have a similar order of convergence (EOC. $\approx O(h^{1.25})$. The absolute error at any particular step size however, is far smaller in the Ford and Simpson method. This can have an impact on any predictive tool (computer program) developed to compare competing methods for numerical efficiency. In section 6.3 we explained that a linear extrapolation of the path for each method was feasible. A more realistic interpretation of each path would be one with a slightly positive curve. When the absolute errors in competing methods are of the same order of magnitude, the error in extrapolating the paths is approximately the same for each method and thus the overall qualitative behaviour is maintained. This can be automated using a computer program.

h	Adams method				Diethelm and Ford			
	error y(1)	KFlops	Time	EOC	error y(1)	KFlops	Time	EOC
1/32	1.86e-02	14.8	0.00		1.60e-02	16.0	0.00	
1/64	6.62e-03	46.0	0.06	1.49	5.85e-03	41.2	0.00	1.45
1/128	2.35e-03	157.4	0.11	1.49	2.12e-03	119.2	0.05	1.47
1/256	8.36e-04	576.8	0.22	1.49	7.59e-04	385.7	0.17	1.48
1/512	2.96e-04	2202.2	0.50	1.50	2.71e-04	1361.2	0.32	1.49
	Ford and Simpson				new method			
1/32	1.27e-03	9.9	0.06		1.86e-02	27.0	0.05	
1/64	4.32e-04	22.8	0.05	1.56	6.62e-03	70.1	0.11	1.49
1/128	1.49e-04	57.7	0.11	1.54	2.35e-03	205.4	0.17	1.49
1/256	5.14e-05	164.6	0.22	1.54	8.36e-04	672.6	0.49	1.49
1/512	1.79e-05	525.6	0.55	1.52	2.96e-04	2393.4	1.21	1.50
	Adams optimum (M=2)				new method, optimum (M=2)			
1/32	4.61e-04	22.1	0.00		4.61e-04	39.0	0.11	
1/64	8.96e-05	68.7	0.06	2.36	8.96e-05	102.4	0.16	2.36
1/128	1.80e-05	235.6	0.16	2.32	1.80e-05	302.8	0.33	2.32
1/256	3.76e-06	864.3	0.28	2.26	3.76e-06	998.4	0.82	2.26
1/512	8.14e-07	3301.4	0.77	2.21	8.14e-07	3569.4	1.87	2.21

Table 8.9: $D^2y + D^{1/2}y + y = t^3 + 6t + \frac{3.2t^{2.5}}{\Gamma(0.5)}$

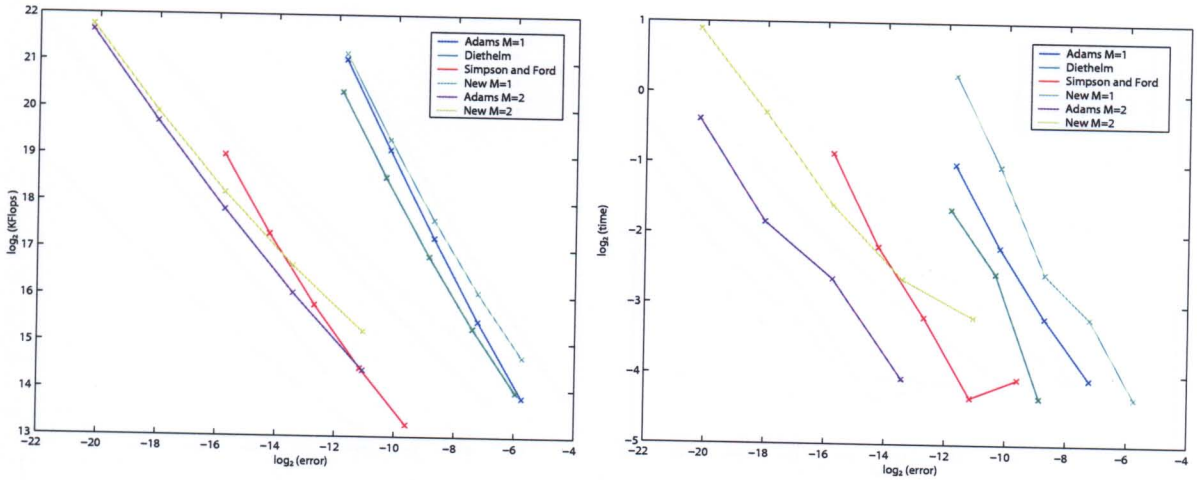

Figure 8.6: $D^2y + D^{1/2}y + y = t^3 + 6t + \frac{3.2t^{2.5}}{\Gamma(0.5)}$

h	Adams method				Diethelm and Ford			
	error y(1)	KFlops	Time	EOC	error y(1)	KFlops	Time	EOC
1/32	4.84e-03	14.8	0.00		1.30e-02	16.0	0.00	
1/64	1.66e-03	45.8	0.06	1.54	7.40e-03	41.1	0.00	0.81
1/128	5.74e-04	157.1	0.11	1.53	4.02e-03	118.9	0.06	0.88
1/256	2.00e-04	576.3	0.22	1.52	2.12e-03	385.2	0.16	0.92
1/512	7.00e-05	2201.2	0.50	1.51	1.10e-03	1360.2	0.38	0.95
	Ford and Simpson				new method			
1/32	9.08e-04	9.8	0.06		4.84e-03	26.9	0.05	
1/64	3.32e-04	22.6	0.05	1.45	1.66e-03	69.9	0.11	1.54
1/128	1.20e-04	57.5	0.11	1.47	5.74e-04	205.1	0.21	1.53
1/256	4.30e-05	164.0	0.22	1.48	2.00e-04	672.0	0.55	1.52
1/512	1.54e-05	524.6	0.60	1.48	7.00e-05	2392.4	1.27	1.51
	Adams optimum (M=2)				new method, optimum (M=2)			
1/32	6.06e-04	22.0	0.06		6.06e-04	39.0	0.05	
1/64	2.73e-04	68.5	0.06	1.15	2.73e-04	102.2	0.16	1.15
1/128	1.13e-04	235.3	0.17	1.27	1.13e-04	302.5	0.33	1.27
1/256	4.46e-05	863.8	0.33	1.34	4.46e-05	997.9	0.77	1.34
1/512	1.70e-05	3300.4	0.77	1.39	1.70e-05	3568.4	1.97	1.39

Table 8.10: $D^2y + D^{1/2}y + y = t^2 + 2 + \frac{2.6666666667t^{1.5}}{\Gamma(0.5)}$

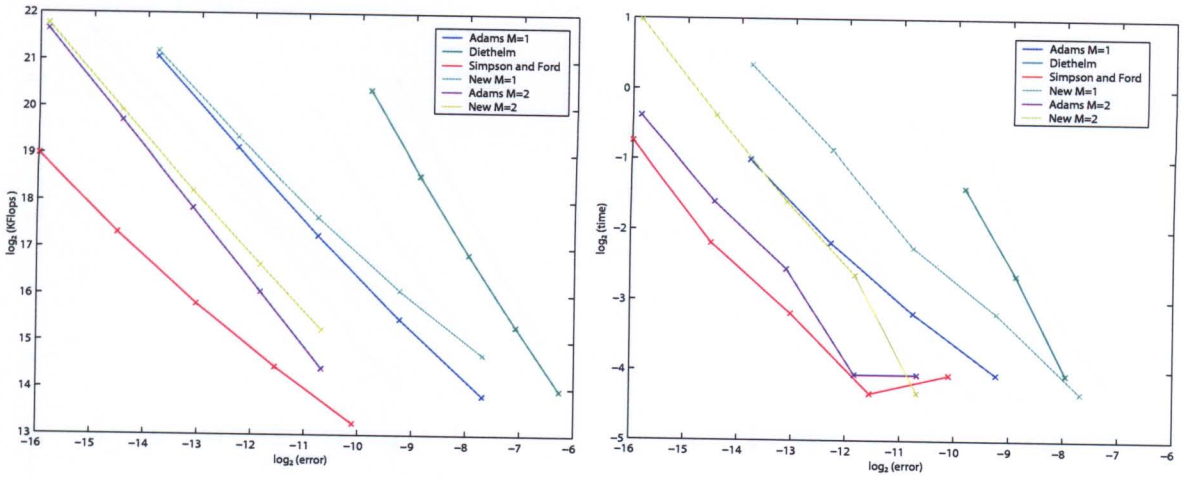

Figure 8.7: $D^2y + D^{1/2}y + y = t^2 + 2 + \frac{2.6666666667t^{1.5}}{\Gamma(0.5)}$

h	Adams method				Diethelm and Ford			
	error y(1)	KFlops	Time	EOC	error y(1)	KFlops	Time	EOC
1/32	1.12e-01	25.3	0.00		7.37e-03	44.7	0.05	
1/64	4.72e-02	83.4	0.06	1.24	2.34e-03	106.6	0.05	1.66
1/128	1.99e-02	297.7	0.11	1.24	7.30e-04	282.8	0.11	1.68
1/256	8.41e-03	1119.6	0.22	1.24	2.26e-04	844.0	0.22	1.69
1/512	3.54e-03	4336.2	0.72	1.25	6.93e-05	2802.0	0.49	1.71
	Ford and Simpson				new method			
1/32	3.86e-03	9.9	0.00		2.07e-02	27.0	0.06	
1/64	1.61e-03	22.8	0.06	1.26	8.20e-03	70.1	0.11	1.34
1/128	6.71e-04	57.7	0.05	1.26	3.29e-03	205.4	0.22	1.32
1/256	2.81e-04	164.6	0.22	1.26	1.33e-03	672.6	0.55	1.31
1/512	1.18e-04	525.6	0.55	1.25	5.43e-04	2393.4	1.21	1.29
	Adams optimum (M=4)				new method, optimum (M=4)			
1/32	1.42e-03	64.0	0.05		1.48e-03	51.1	0.11	
1/64	3.07e-04	209.9	0.11	2.21	3.62e-04	134.7	0.22	2.03
1/128	6.62e-05	747.3	0.28	2.21	8.75e-05	400.2	0.49	2.05
1/256	1.43e-05	2805.3	0.60	2.21	2.11e-05	1324.3	1.10	2.05
1/512	3.10e-06	10853.4	1.81	2.21	5.07e-06	4745.5	2.53	2.06

Table 8.11: $D^2y + D^{3/4}y + y = t^3 + 6t + \frac{8.533333333t^{2.25}}{\Gamma(0.25)}$

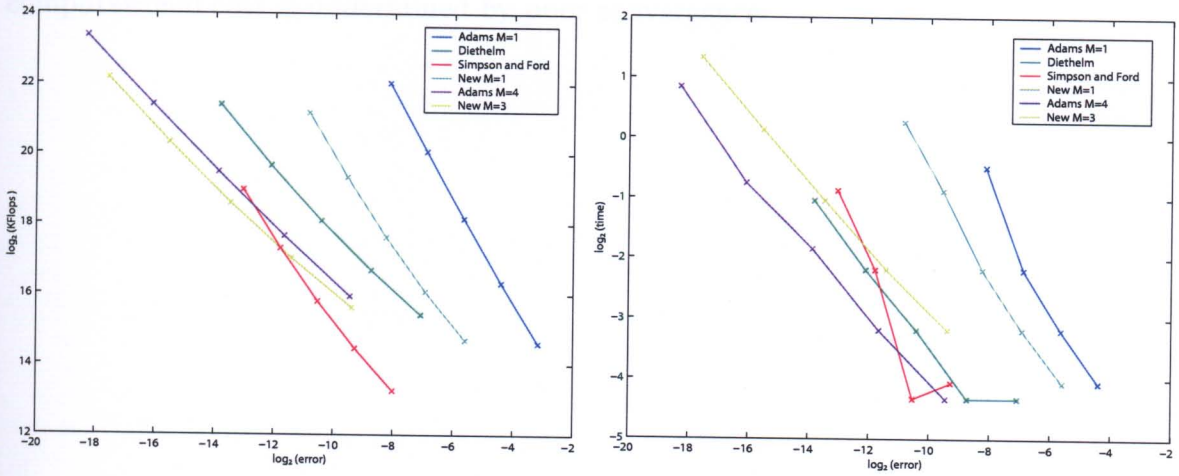

Figure 8.8: $D^2y + D^{3/4}y + y = t^3 + 6t + \frac{8.533333333t^{2.25}}{\Gamma(0.25)}$

8.5.4 Case 3: $D^2y(t) + b_1D^\alpha y(t) + b_0Dy(t) + a_0y(t) = g(t)$

There are three test equations used for experimentation in this section:

(a) $D^2y + D^{3/2}y + y = t^3 + 6t + \frac{8t^{1.5}}{\Gamma(0.5)}$, where $y(0) = 0, y'(0) = 0$.

(b) $D^2y + D^{3/2}y + y = t^2 + 2 + \frac{4t^{0.5}}{\Gamma(0.5)}$, where $y(0) = 0, y'(0) = 0$.

(c) $D^2y + D^{7/4}y + y = t^3 + 6t + \frac{19.2t^{1.25}}{\Gamma(0.25)}$, where $y(0) = 0, y'(0) = 0$.

Equations (a) and (c) have exact solutions $y = t^3$, equation (b) has an exact solution $y = t^2$.

In table 8.12 and the corresponding figure 8.9 we show that for equation (a) the Adams method (M=2) is most economical, both when measuring using KFlops or by time. Test equation (b) highlights a special case for the Ford and Simpson method, in which the error for all step sizes is zero. As we can see from table 8.13 and figure 8.10 the Adams method (M=2) is the next most efficient, but as $D^{7/4}(t^2) \in C[0, T]$ only achieves the EOC of $O(h^{1.5})$.

From table 8.14 and figure 8.11 it is clear that the new method is the most economical, both in terms of Flops and time for equation (c). We note that for this equation the new method requires a system of 4 equations to be created, as does the Ford and Simpson method, whereas the Diethelm and Adams methods require an 8 equation system. This results in a substantial saving in computational cost for both the new and Ford and Simpson methods. As the Ford and Simpson method's order of convergence is $O(h^{1+\lceil\alpha\rceil-\alpha}) = O(h^{1.25})$, the saving in computational cost is undermined by poor convergence.

h	Adams method				Diethelm and Ford			
	error y(1)	KFlops	Time	EOC	error y(1)	KFlops	Time	EOC
1/32	2.06e-02	14.8	0.06		1.75e-02	16.0	0.00	
1/64	7.40e-03	46.0	0.06	1.47	6.40e-03	41.2	0.05	1.45
1/128	2.65e-03	157.4	0.11	1.48	2.31e-03	119.2	0.06	1.47
1/256	9.44e-04	576.8	0.22	1.49	8.29e-04	385.8	0.16	1.48
1/512	3.36e-04	2202.2	0.49	1.49	2.96e-04	1361.2	0.39	1.49
	Ford and Simpson				new method			
1/32	2.47e-03	10.0	0.00		2.06e-02	27.0	0.06	
1/64	8.51e-04	23.0	0.00	1.53	7.40e-03	70.1	0.11	1.47
1/128	2.96e-04	58.1	0.11	1.53	2.65e-03	205.4	0.22	1.48
1/256	1.03e-04	165.3	0.22	1.52	9.44e-04	672.6	0.50	1.49
1/512	3.61e-05	527.2	0.54	1.51	3.36e-04	2393.4	1.21	1.49
	Adams optimum (M=2)				new method, optimum (M=2)			
1/32	8.78e-04	22.1	0.06		8.78e-04	39.0	0.05	
1/64	2.01e-04	68.7	0.06	2.13	2.01e-04	102.4	0.16	2.13
1/128	4.71e-05	235.6	0.16	2.09	4.71e-05	302.8	0.33	2.09
1/256	1.12e-05	864.3	0.33	2.07	1.12e-05	998.4	0.77	2.07
1/512	2.71e-06	3301.5	0.77	2.05	2.71e-06	3569.4	1.81	2.05

Table 8.12: $D^2y + D^{3/2}y + y = t^3 + 6t + \frac{8t^{1.5}}{\Gamma(0.5)}$

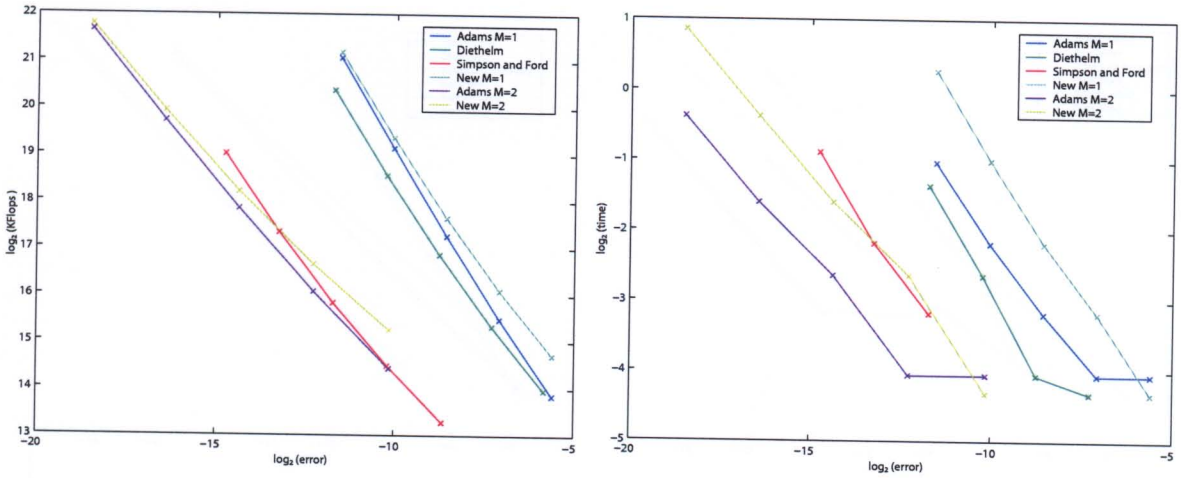

Figure 8.9: $D^2y + D^{3/2}y + y = t^3 + 6t + \frac{8t^{1.5}}{\Gamma(0.5)}$

h	Adams method				Diethelm and Ford			
	error y(1)	KFlops	Time	EOC	error y(1)	KFlops	Time	EOC
1/32	9.51e-03	14.8	0.00		8.31e-03	16.0	0.00	
1/64	3.37e-03	45.8	0.06	1.50	5.13e-03	41.1	0.00	0.70
1/128	1.19e-03	157.1	0.11	1.50	2.92e-03	119.0	0.06	0.81
1/256	4.21e-04	576.3	0.22	1.50	1.59e-03	385.2	0.17	0.88
1/512	1.49e-04	2201.2	0.50	1.50	8.41e-04	1360.2	0.33	0.92
	Ford and Simpson				new method			
1/32	0	2.3	0.00		9.51e-03	26.9	0.05	
1/64	0	4.7	0.06		3.37e-03	70.0	0.11	1.50
1/128	0	10.0	0.11		1.19e-03	205.1	0.22	1.50
1/256	0	23.0	0.22		4.21e-04	672.0	0.49	1.50
1/512	0	58.3	0.55		1.49e-04	2392.4	1.15	1.50
	Adams optimum (M=2)				new method, optimum (M=2)			
1/32	2.25e-03	22.0	0.00		2.25e-03	39.0	0.11	
1/64	8.10e-04	68.6	0.11	1.47	8.10e-04	102.2	0.16	1.47
1/128	2.91e-04	235.3	0.11	1.48	2.91e-04	302.5	0.38	1.48
1/256	1.04e-04	863.8	0.27	1.48	1.04e-04	997.9	0.77	1.48
1/512	3.73e-05	3300.4	0.72	1.48	3.73e-05	3568.4	1.70	1.48

Table 8.13: $D^2y + D^{3/2}y + y = t^2 + 2 + \frac{4t^{0.5}}{\Gamma(0.5)}$

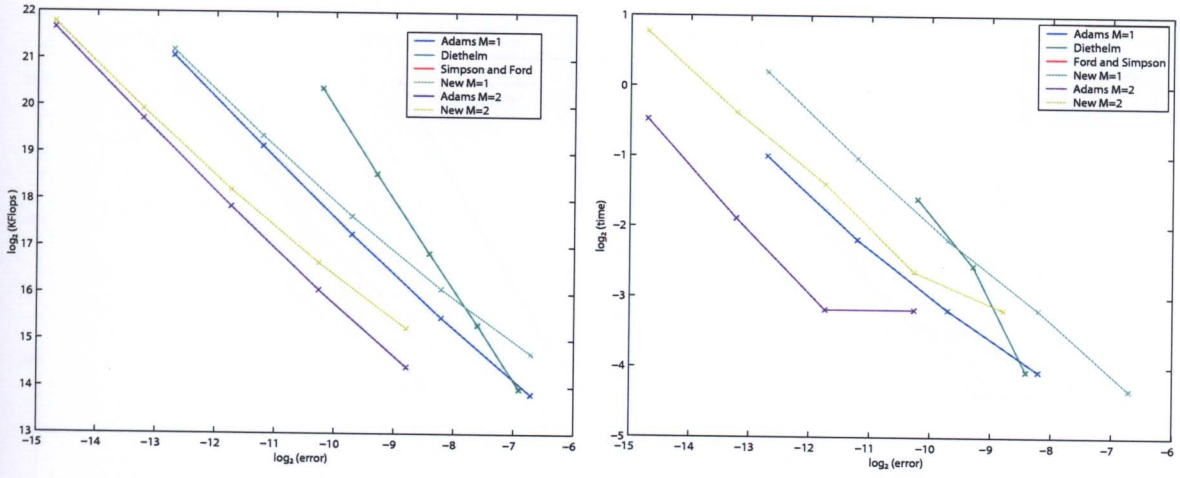

Figure 8.10: $D^2y + D^{3/2}y + y = t^2 + 2 + \frac{4t^{0.5}}{\Gamma(0.5)}$

h	Adams method				Diethelm and Ford			
	error y(1)	KFlops	Time	EOC	error y(1)	KFlops	Time	EOC
1/32	1.25e-01	25.4	0.00		8.78e-03	44.7	0.05	
1/64	5.39e-02	83.4	0.06	1.22	2.79e-03	106.7	0.05	1.66
1/128	2.29e-02	297.7	0.06	1.23	8.73e-04	282.8	0.11	1.68
1/256	9.70e-03	1119.6	0.27	1.24	2.70e-04	844.0	0.17	1.69
1/512	4.10e-03	4336.2	0.60	1.24	8.31e-05	2802.0	0.44	1.70
	Ford and Simpson				new method			
1/32	7.52e-03	10.0	0.00		1.57e-03	27.0	0.06	
1/64	3.14e-03	23.0	0.06	1.26	3.19e-04	70.1	0.11	2.30
1/128	1.31e-03	58.1	0.11	1.26	4.42e-04	205.4	0.22	0.47
1/256	5.51e-04	165.3	0.22	1.25	2.88e-04	672.6	0.55	0.62
1/512	2.31e-04	527.2	0.55	1.25	1.56e-04	2393.4	1.26	0.88
	Adams optimum (M=4)				new method, optimum (M=4)			
1/32	2.73e-03	64.0	0.10		5.45e-04	63.2	0.11	
1/64	6.64e-04	209.9	0.11	1.50	1.36e-04	167.0	0.27	2.00
1/128	1.61e-04	747.3	0.22	1.69	3.50e-05	497.6	0.55	1.96
1/256	3.89e-05	2805.3	0.60	2.05	9.13e-06	1650.2	1.32	1.94
1/512	9.43e-06	10853.5	1.60	2.04	2.38e-06	5921.6	3.13	1.94

Table 8.14: $D^2y + D^{7/4}y + y = t^3 + 6t + \frac{19.2t^{1.25}}{\Gamma(0.25)}$

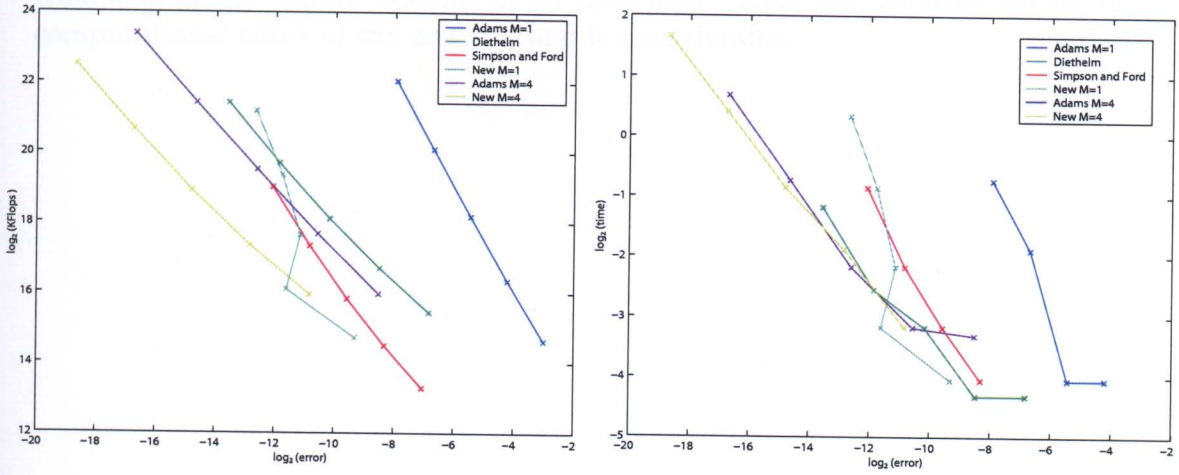

Figure 8.11: $D^2y + D^{7/4}y + y = t^3 + 6t + \frac{19.2t^{1.25}}{\Gamma(0.25)}$

8.5.5 Case 4: $D^2y(t)+b_{N-1}D^{\alpha_{N-1}}y(t)+\ldots+b_1D^{\alpha_1}y(t)+b_0y(t) = g(t)$

We now give examples of FDEs with multiple fractional components. There are two test equations explored in this section:

(a) $D^2y + D^{3/2}y + Dy + D^{1/2}y + y = t^3 + 3t^2 + 6t + \frac{8t^{3/2}}{\Gamma(0.5)} + \frac{3.2t^{5/2}}{\Gamma(0.5)}$, where $y(0) = 1, y'(0) = 0$.

(b) $D^2y+D^{3/2}y+Dy+D^{1/5}y+y = t^3+3t^2+6t+\frac{8t^{1.5}}{\Gamma(0.5)}+\frac{1.488095238t^{14/5}}{\Gamma(0.8)}$, where $y(0) = 0, y'(0) = 0$.

Equations (a) and (b) have exact solutions $y = t^3$.

Figure 8.12, shows that the Adams method (M=2) is the most numerically efficient both in KFlops and in time measurements. The paths associated with the Adams method (M=2) and the new method (M=2) are erratic. These paths stabilize as the step size reduces. When implementing a computer program to discover the optimum method, we should therefore test whether the paths are stable for our chosen step sizes, and adjust accordingly. The reader should be aware, as stated earlier for a different example, that the Adams method and the new method for this equation, implement exactly the same algorithm, and the difference in performance is due to programming issues.

In table 8.16 and associated figure 8.13, the new method with (M=3) is the most numerically efficient, both in measuring by KFlops or time. In this example the new method requires a system of 4 equations, whereas the Diethelm and Adams methods require systems of 20 equations. Thus the resulting saving in computational effort of the new method is considerable.

h	Adams method				Diethelm and Ford			
	error y(1)	KFlops	Time	EOC	error y(1)	KFlops	Time	EOC
1/8	1.61e-02	15.1	0.00		1.48e-02	16.3	0.06	
1/16	5.76e-03	46.5	0.00	1.48	5.39e-03	41.8	0.05	1.46
1/32	2.05e-03	158.5	0.06	1.49	1.94e-03	120.3	0.11	1.47
1/64	7.30e-04	578.9	0.22	1.49	6.97e-04	387.8	0.16	1.48
1/128	2.59e-04	2206.3	0.55	1.49	2.49e-04	1365.4	0.33	1.49
	Ford and Simpson				new method			
1/8	2.44e-03	17.8	0.00		1.61e-02	27.3	0.05	
1/16	8.46e-04	41.7	0.11	1.53	5.76e-03	70.6	0.06	1.48
1/32	2.95e-04	107.8	0.11	1.52	2.05e-03	206.4	0.21	1.49
1/64	1.03e-04	313.7	0.33	1.52	7.30e-04	674.7	0.55	1.49
1/128	3.62e-05	1020.6	0.88	1.51	2.59e-04	2397.5	1.21	1.49
	Adams optimum (M=2)				new method, optimum (M=2)			
1/8	1.38e-04	22.4	0.06		1.38e-04	39.4	0.11	
1/16	9.00e-06	69.2	0.11	3.94	9.00e-06	102.9	0.16	3.94
1/32	2.00e-06	236.7	0.17	2.17	2.00e-06	303.8	0.33	2.17
1/64	1.22e-06	866.4	0.28	0.71	1.22e-06	1000.5	0.71	0.71
1/128	4.28e-07	3305.6	0.77	1.51	4.28e-07	3573.6	1.86	1.51

Table 8.15: $D^2y + D^{3/2} + Dy + D^{1/2}y + y = t^3 + 3t^2 + 6t + \frac{8t^{3/2}}{\Gamma(0.5)} + \frac{3.2t^{5/2}}{\Gamma(0.5)}$

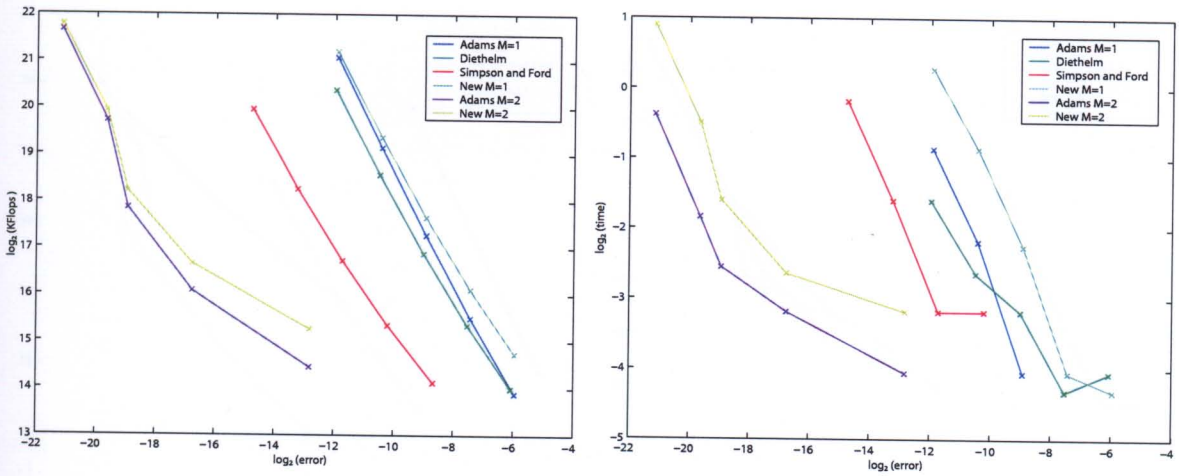

Figure 8.12: $D^2y + D^{3/2} + Dy + D^{1/2}y + y = t^3 + 3t^2 + 6t + \frac{8t^{3/2}}{\Gamma(0.5)} + \frac{3.2t^{5/2}}{\Gamma(0.5)}$

h	Adams method				Diethelm and Ford			
	error y(1)	KFlops	Time	EOC	error y(1)	KFlops	Time	EOC
1/32	4.41e-01	57.1	0.05		4.49e-03	226.0	0.06	
1/64	2.14e-01	196.0	0.05	1.04	1.32e-03	493.9	0.11	1.77
1/128	1.01e-01	719.6	0.11	1.08	3.82e-04	1155.6	0.17	1.79
1/256	4.77e-02	2749.8	0.28	1.08	1.10e-04	2982.7	0.38	1.80
1/512	2.23e-02	10742.4	0.83	1.10	3.12e-05	8652.2	0.88	1.82
	Ford and Simpson				new method			
1/32	2.01e-03	17.8	0.00		2.71e-02	27.3	0.05	
1/64	6.83e-04	41.7	0.06	1.56	1.14e-02	70.6	0.11	1.25
1/128	2.34e-04	107.8	0.16	1.54	4.82e-03	206.4	0.17	1.24
1/256	8.08e-05	313.7	0.28	1.53	2.06e-03	674.7	0.49	1.23
1/512	2.81e-05	1020.6	0.93	1.52	8.88e-04	2397.5	1.10	1.21
	Adams optimum (M=11)				new method, optimum (M=3)			
1/32	2.51e-04	354.8	0.17		7.55e-04	51.4	0.11	
1/64	5.23e-05	1201.3	0.27	2.26	2.13e-04	135.3	0.22	1.83
1/128	4.08e-05	4368.7	0.71	0.36	5.75e-05	401.2	0.49	1.89
1/256	1.67e-05	16601.8	1.82	1.29	1.52e-05	1326.4	0.98	1.92
1/512	5.65e-06	64660.9	5.60	1.56	3.95e-06	4749.7	2.53	1.94

Table 8.16: $D^2y + D^{3/2}y + Dy + D^{1/5}y + y = t^3 + 3t^2 + 6t + \frac{8t^{1.5}}{\Gamma(0.5)} + \frac{1.488095238t^{14/5}}{\Gamma(0.8)}$

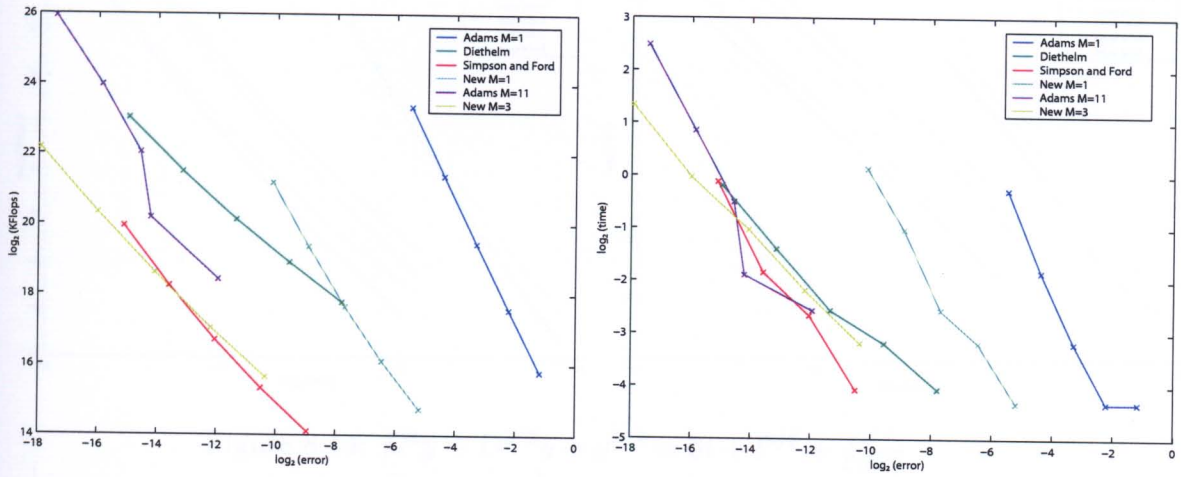

Figure 8.13: $D^2y + D^{3/2}y + Dy + D^{1/5}y + y = t^3 + 3t^2 + 6t + \frac{8t^{1.5}}{\Gamma(0.5)} + \frac{1.488095238t^{14/5}}{\Gamma(0.8)}$

8.5.6 Case 5: $D^{\alpha_N} y(t) = f(t, y(t), D^{\alpha_{N-1}} y(t), \ldots, D^{\alpha_1} y(t))$,

There are currently only two methods suitable for calculating the solution of non-linear multi-term FDEs, the Adams method (see section 7.2.2) and the new method (see section 7.4). For certain equations (e.g. equation (a)) these two methods are identical, and the difference in performance is due to programming issues. For such equations the Adams method always produces better numerical efficiency.

There are 2 test equations investigated in this section:

(a) $D^2 y + D^{3/2} y + y^{3/2} = 6t + t^{9/2} + \frac{8t^{1.5}}{\Gamma(0.5)}$, exact answer $y = t^3$.

(b) $D^2 y + t^2 D^{1.4} y + y^{3/2} = 6t + t^{9/2} + \frac{6.25t^{3.6}}{\Gamma(0.6)}$, exact answer $y = t^3$.

For equation (a), in table 8.17 and Figure 8.14, we discover that the optimum number of correctors for each method is 2.

For equation (b) the Adams method produces a system of 20 equations and requires 6 corrector iterations for optimum numerical efficiency. The new method only requires a system of 4 equations and 3 corrector iterations for optimum numerical efficiency. In table 8.18 and corresponding figure 8.15, we show that this results in the new method reducing the time/KFlops required for a given accuracy considerably.

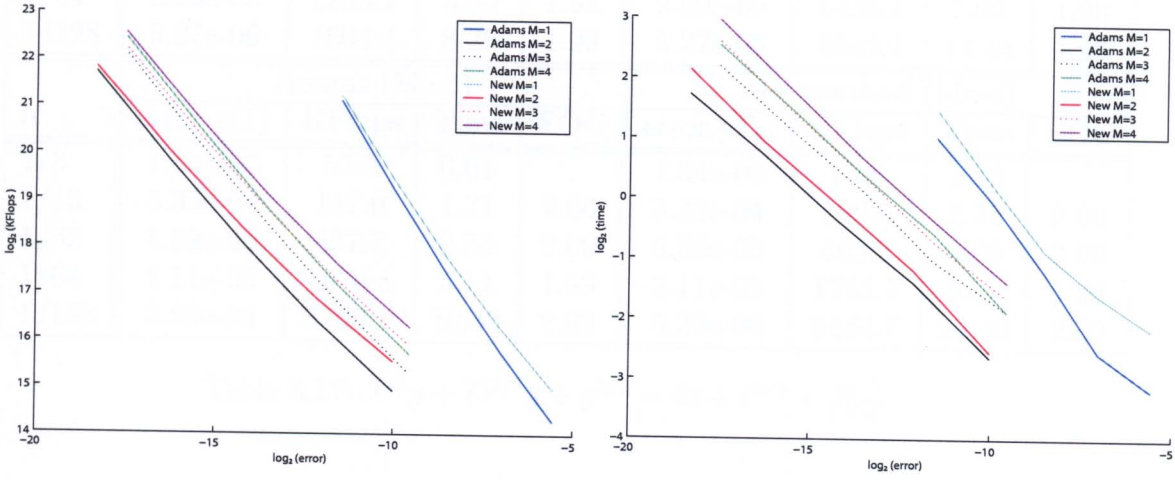

Figure 8.14: $D^2 y + D^{3/2} y + y^{3/2} = 6t + t^{9/2} + \frac{8t^{1.5}}{\Gamma(0.5)}$

h	Adams (M=1)				new method (M=1)			
	error y(1)	KFlops	Time	EOC				
1/8	2.12e-02	20.1	0.17		2.12e-02	32.2	0.38	
1/16	7.62e-03	56.5	0.44	1.48	7.62e-03	80.6	0.77	1.48
1/32	2.73e-03	178.6	0.88	1.48	2.73e-03	226.6	1.59	1.48
1/64	9.71e-04	619.3	1.76	1.49	9.71e-04	715.0	3.35	1.49
1/128	3.45e-04	2287.1	4.01	1.49	3.45e-04	2478.3	6.75	1.49
h	Adams (M=2)				new method (M=2)			
	error y(1)	KFlops	Time	EOC	error y(1)	KFlops	Time	EOC
1/8	9.67e-04	31.1	0.38		9.67e-04	48.0	0.61	
1/16	2.22e-04	86.7	0.71	2.12	2.22e-04	120.4	1.26	2.12
1/32	5.22e-05	271.6	1.37	2.09	5.22e-05	338.8	2.58	2.09
1/64	1.24e-05	936.5	2.96	2.07	1.24e-05	1070.6	5.28	2.07
1/128	3.03e-06	3445.8	6.37	2.03	3.00e-06	3713.8	10.72	2.03
	Adams (M=3)				new method (M=3)			
1/8	1.25e-03	42.0	0.49		1.25e-03	63.8	0.82	
1/16	3.24e-04	116.8	0.99	1.95	3.24e-04	160.1	1.81	1.95
1/32	8.26e-05	364.7	1.92	1.97	8.26e-05	451.0	3.52	1.97
1/64	2.09e-05	1253.7	3.90	1.98	2.09e-05	1426.2	7.03	1.98
1/128	5.27e-06	4604.4	8.29	1.99	5.27e-06	4949.2	14.44	1.99
h	Adams (M=4)				new method (M=4)			
	error y(1)	KFlops	Time	EOC	error y(1)	KFlops	Time	EOC
1/8	1.34e-03	53.0	0.61		1.34e-03	79.6	1.10	
1/16	3.35e-04	147.0	1.21	2.00	3.35e-04	199.9	2.15	2.00
1/32	8.39e-05	457.7	2.53	2.00	8.39e-05	563.3	4.39	2.00
1/64	2.11e-05	1570.8	5.11	1.99	2.11e-05	1781.7	9.39	1.99
1/128	5.29e-06	5763.1	10.33	2.00	5.29e-06	6184.7	18.40	2.00

Table 8.17: $D^2y + D^{3/2}y + y^{3/2} = 6t + t^{9/2} + \frac{8t^{1.5}}{\Gamma(0.5)}$

h	Adams (M=1)				new method (M=1)			
	error y(1)	KFlops	Time	EOC				
1/32	1.95e-01	36.1	0.22		2.70e-02	32.4	0.11	
1/64	8.68e-02	113.0	0.44	1.17	1.00e-02	81.0	0.27	1.43
1/128	3.82e-02	389.8	0.88	1.18	3.71e-03	227.3	0.60	1.43
1/256	1.67e-02	1434.9	1.81	1.19	1.37e-03	716.5	1.26	1.44
1/512	7.31e-03	5491.3	3.84	1.19	5.05e-04	2481.4	2.69	1.44
h	Adams (M=5)				new method (M=2)			
	error y(1)	KFlops	Time	EOC	error y(1)	KFlops	Time	EOC
1/32	3.42e-03	114.2	0.77		1.86e-03	48.3	0.27	
1/64	8.47e-04	351.3	1.53	2.01	4.84e-04	121.0	0.50	1.94
1/128	2.07e-04	1194.1	3.14	2.03	1.28e-04	340.1	0.94	1.92
1/256	5.03e-05	4354.3	6.48	2.04	3.42e-05	1073.1	1.98	1.90
1/512	1.22e-05	16573.0	13.57	2.04	9.19e-06	3718.9	4.51	1.90
	Adams (M=6)				new method (M=3)			
1/32	3.97e-04	133.8	0.94		1.10e-03	64.2	0.28	
1/64	2.23e-04	410.9	1.82	0.83	2.93e-04	161.0	0.66	1.91
1/128	8.23e-05	1395.2	3.68	1.44	7.61e-05	452.8	1.37	1.94
1/256	2.63e-05	5084.2	7.64	1.65	1.95e-05	1429.7	2.75	1.96
1/512	7.80e-06	19343.4	16.42	1.75	4.97e-06	4956.4	6.04	1.97
h	Adams (M=7)				new method (M=4)			
	error y(1)	KFlops	Time	EOC	error y(1)	KFlops	Time	EOC
1/8	1.96e-03	153.3	1.04		1.33e-03	80.1	0.44	
1/16	5.84e-04	470.5	2.14	1.75	3.30e-04	201.0	0.82	2.01
1/32	1.64e-04	1596.3	4.28	1.83	8.25e-05	565.6	1.65	2.00
1/64	4.46e-05	5814.0	8.84	1.88	2.07e-05	1786.3	3.52	1.99
1/128	1.19e-05	22113.8	18.73	1.91	5.19e-06	6193.9	7.63	2.00

Table 8.18: $D^2y + t^2D^{1.4}y + y^{3/2} = 6t + t^{9/2} + \frac{6.25t^{3.6}}{\Gamma(0.6)}$

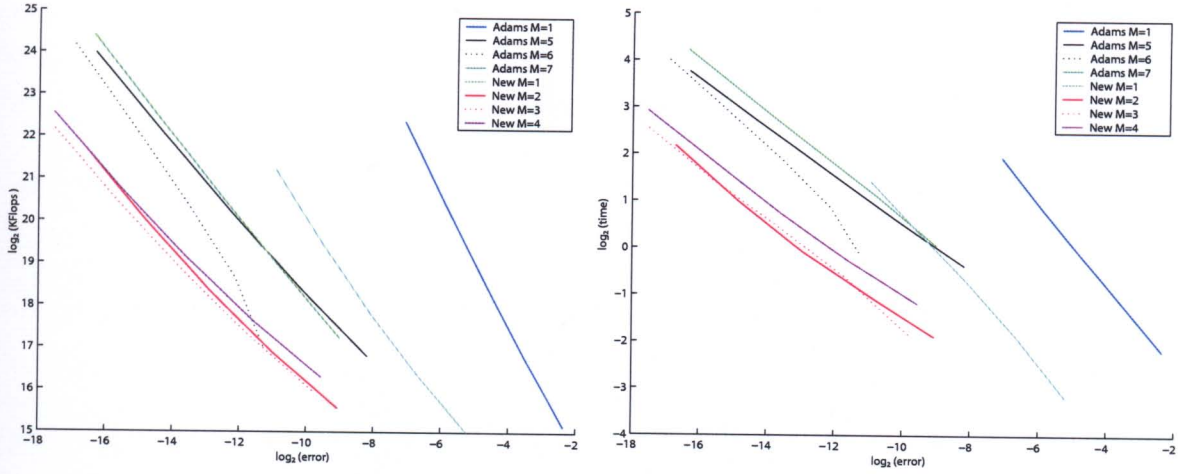

Figure 8.15: $D^2y + t^2D^{1.4}y + y^{3/2} = 6t + t^{9/2} + \frac{6.25t^{3.6}}{\Gamma(0.6)}$

8.5.7 Estimating Run Time

So far we have concentrated on describing the performance of FDE numerical methods with regard to step size. In real world applications, scientists and mathematicians are likely to want the solution of a FDE, with a given error tolerance, in the quickest time possible.

We can use the graphical technique introduced in section 6.3 to estimate the run time of competing methods.

Figure 8.16 illustrates how we can use a simple extrapolation procedure to give an estimate of run time.

We let the exact analytical solution, for a fixed interval T be $y(T)$ and the approximation at T using n steps be $y_n(T)$. Let the order of convergence of the given method be p. We now let the run time for n steps be RT_n.

For a given error tolerance of ET the estimated run time RT_{est} is calculated using

$$RT_{est} = 2^{(grad \times log_2(ET) + const)} \tag{8.7}$$

where,

$$grad = \frac{\log_2(RT_n) - \log_2(RT_{2n})}{\log_2(|(y(T) - y_n(T)|) - \log_2(|(y(T) - y_{2n}(T)|)}$$

and

$$const = \log_2(RT_n) - grad \times \log_2((|(y(T) - y_n(T)|).$$

We can also estimate the number of steps $\tilde{n}$, over the interval T, required for a given accuracy,

$$\tilde{n} = 2n\frac{|(y(T) - y_{2n}(T)|}{ET^{1/p}}. \tag{8.8}$$

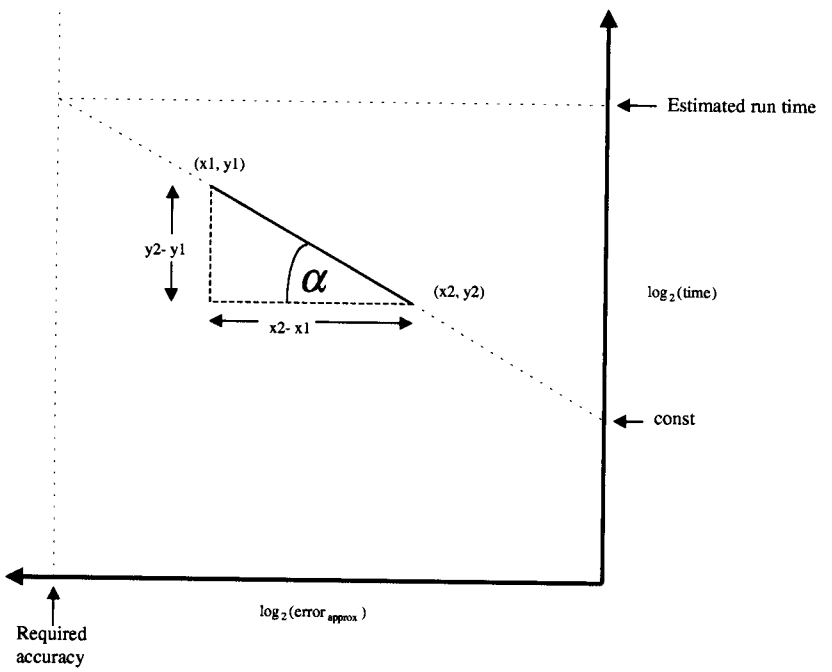

Figure 8.16: calculating the gradient and constant

Consider the two test equations,

$$D^2y + D^{3/2}y + y = t^3 + 6t + \frac{8t^{1.5}}{\Gamma(0.5)}, \tag{8.9}$$

and

$$D^2y + D^{1.6}y + y = t^3 + 6t + \frac{10.71428571t^{1.4}}{\Gamma(0.4)}, \tag{8.10}$$

where $y(0) = 0, y'(0) = 0$, both with an exact solution of $y = t^3$.

We use formula (8.7) to estimate the run time required for a given accuracy and calculate the actual run time required using $\tilde{n}$ steps, obtained from equation (8.8). Table 8.19 illustrates how the run time estimate can vary depending upon the number of steps n. We can see that the run time estimate is more accurate for a large number of steps and that sometimes the estimate of run time is far too low. The estimate of run time however gives good relative behaviour of the methods. We can therefore determine which method is the most numerically efficient. The actual run time of any particular method varies only by a small percentage, this is due to the asymptotic nature of the order of convergence. Experimentation on wide selection of equations has concluded that in practice a relatively small value of n is required to give an accurate estimate of $\tilde{n}$. We can therefore predict which method is most numerically efficient and obtain solutions with the required degree of accuracy.

When more than two data points have been calculated, it may be possible in the future to obtain more accurate estimates of actual run time by extrapolating the data represented in figure 8.16 by the points (x1,y1,x2,y2), with a curve instead of a straight line. We could achieve this by applying curve fitting algorithms to the $log_2(error)$ v $log_2(time)$ data. This however is beyond the scope of the thesis and will not be explored further.

8.6 Conclusions

In this chapter we have shown how the optimum number of corrector iterations for the Adams method (see section 7.2.2) and for the proposed new method (see Section 7.4) vary depending upon the order of constituent fractional derivatives. The new algorithm is then compared with the other available methods for various classes of multi-term FDEs (see tables 8.7 to 8.18).

We then showed how we could obtain a specific degree of accuracy for a given equation at a given y value and gave a method for obtaining a rough estimate of run time.

n	Time required to achieve desired accuracy of 1e-06 (estimate in brackets) for equation $D^2y + D^{3/2}y + y = t^3 + 6t + \frac{8t^{1.5}}{\Gamma(0.5)}$			
	Diethelm	Adams	new method	Ford and Simpson
1024	(0:38) 3:52	(0:01) 0:01	(0:03) 0:03	(0:20) 0:25
2048	(0:44) 3:53	(0:01) 0:01	(0:03) 0:03	(0:26) 0:25
4096	(1:35) 3:56	(0:01) 0:01	(0:03) 0:03	(0:25) 0:25
8192	(2:53) 4:06	(0:01) 0:01	(0:03) 0:03	(0:25) 0:25
16384	(3:27) 4:07	(0:01) 0:01	(0:03) 0:03	(0:25) 0:25
n	Time required to achieve desired accuracy of 1e-07 (estimate in brackets) for the equation $D^2y + D^{1.6}y + y = t^3 + 6t + \frac{10.71428571t^{1.4}}{\Gamma(0.4)}$			
	Diethelm	Adams	new method	Ford and Simpson
1024	(0:23) 4:10	(0:51) 1:39	(0:26) 0:44	(3:46) 11:43
2048	(1:12) 4:19	(1:29) 1:53	(0:37) 0:49	(4:14) 13:58
4096	(2:27) 4:35	(2:05) 2:09	(0:55) 0:56	(6:50) 20:04
8192	(3:40) 4:40	(2:21) 2:20	(1:06) 1:03	(12:17) 21:16

Table 8.19: time required for desired accuracy

Linear Equations

We discovered that the most numerically efficient method depends primarily upon the orders of the multi-term FDEs derivatives. In section 7.2 we showed that the Diethelm and Ford scheme [18] produces a system of equations of size $1/q$, where q is the greatest common divisor of a particular multi-term equation's derivatives. The size of the system has a direct impact upon the numerical efficiency of the method. The Diethelm and the Adams methods always produce the same size of system, which is always greater than or equal to the size of systems produced by the Ford and Simpson method and the new method.

We demonstrated that the order of convergence of the Diethelm and the Ford and Simpson methods vary depending upon the order of the equation's derivatives, and that the Adams method and the new method can achieve order of convergence $O(h^2)$ with sufficient corrector iterations (see section 8.1). The optimum number of iterations varies however, depending upon the order of the equation's derivatives.

We discovered that the differentiability of the function $g(t)$ can effect which numerical method is best (see section 8.5.4), and illustrated that as step size decreases, the most numerically efficient method can change. In section 8.2 it was shown how the stability of both the Adams method and the new method depend upon any constants involved. In certain equations the step size required for stability becomes computationally prohibitive.

For the reasons stated above, it is frequently not apparent which method can be considered most numerically efficient.

The new graphical technique introduced in section 6.3 allows a quick comparison of vying numerical methods to be performed and rough estimates of run-time to be given.

In chapter 12 we show the schematics of a new FDE 'black box' solver, which can automate the comparison of multi-term numerical methods and obtain graphical and data solutions to any given multi-term FDE.

Non-Linear Equations

The new method is the most numerically efficient for all but the set of equations where the Adams method and the new method produce the same results. This happens when the new method and the Adams method produce the same system of equations. In this case the algorithms are structurally identical, but the Adams method is more economical due to programming considerations.

In chapter 9 we describe the concept of a distributed order FDE, that can be thought of as a generalisation of multi-term FDEs.

Chapter 9

Distributed Order Differential Equations

In chapter 7 on multi-term equations, we explained that when a complex process involving several mechanisms is modelled, we use fractional derivatives of several orders to describe the desired behaviour. The concept of a distributed order differential equation (DODE) can be thought of as generalizing a multi-term equation.

In this chapter we give an exposition of the only current method for numerically approximating the solution of a DODE presented by Diethelm and Ford in [14]. We show how the 'framework' can be used to solve DODEs using the Diethelm and Ford's multi-term method [17], introduced by us in chapter 7. The Adams method and the Diethelm method from chapter 7 are used to obtain approximate solutions to a test DODE and the numerical efficiency of the two are compared.

We then show how we can modify Diethelm and Ford's 'framework' and produce an alternative scheme that has possible future benefits.

9.1 Diethelm and Ford Framework

A distributed order differential equation has the form,

$$\sum_{n=1}^{m} \lim_{\epsilon_n \to 0} \int_{\epsilon_n}^{1} b_n(\alpha) I^{\alpha} u^{(n)}(t) d\alpha + \sum_{n=0}^{m} c_n u^{(n)}(t) = g(t). \tag{9.1}$$

Diethelm and Ford [14] introduce a 'basic framework' for the numerical solution of a subclass of equation (9.1), of the form,

$$\int_{a}^{b} A(z) D^{m+z} y(t) dz = g(t). \tag{9.2}$$

The 'framework' involves 2 stages:

1. The integral in equation 9.2 is approximated by a quadrature formula. This results in a linear multi-term equation of the form (7.3).
2. A suitable multi-term numerical method from chapter 7 is applied.

Stage 1

We introduce the quadrature formula

$$\int_a^b \phi(z)dz \approx \sum_{i=0}^{N} w_i \phi(z_i), \tag{9.3}$$

with nodes $z_i \in [a, b]$ and quadrature weights w_i. The function ϕ is given by $\phi(z) = A(z)D^{m+z}y(t)$, with fixed m and t. Diethelm and Ford then replace the integral in equation 9.2 by the approximation

$$\sum_{i=0}^{N} w_i A(z_i) D^{m+z_i} \tilde{y}(t) = g(t).$$

Applying the trapezoidal quadrature formula

$$Q_{N+1}^{Tr}[\phi] = H \sum_{i=0}^{N}{}'' \phi(\alpha_i), \tag{9.4}$$

where H is the step size and the notation $''$ states that where $i = 0$ and $i = N$ the summand should be divided by 2, gives

$$H \sum_{i=0}^{N}{}'' A(z_i) D^{m+z_i} \tilde{y}(t) = g(t). \tag{9.5}$$

Take for example the equation

$$\int_{0.1}^{0.9} \Gamma(3-\alpha) D^{\alpha} y(t) d\alpha = 2\frac{t^{1.9} - t^{1.1}}{\ln t}, \text{ y(0)=0,} \tag{9.6}$$

that has an exact solution of $y = t^2$. This is the test equation used in Diethelm and Ford [14].

Applying the trapezoidal quadrature formula as described above, the test equation becomes,

$$H \sum_{i=0}^{N}{}'' \Gamma(3-\alpha_i) D^{\alpha_i} \tilde{y}_N(t) = 2\frac{t^{1.9} - t^{1.1}}{\ln t}. \tag{9.7}$$

Stage 2

We now choose the multi-term numerical method which is most appropriate. The Ford and Simpson method from section 7.3 and the new method from Section 7.4 are more suited for multi-term FDEs where the largest derivative $b > 1$, which we will not explore in this thesis.

Diethelm and Ford suggest using either the implicit quadrature method introduced in section 5.1, or the predictor corrector method introduced in section 7.2.2. We will investigate the merits of each.

The numerical scheme produces two sources of error; the first from the discretization of the distributed order (M1), and the second from the approximation of the multi-term equation (M2).

It is well established [3], that the trapezoidal quadrature formula generates an error of,

$$Q_{N+1}^{Tr}[\phi] - \int_a^b \phi(z)dz = O(H^2), \tag{9.8}$$

whenever $\phi \in C^2[a, b]$.

Diethelm [11] and Diethelm and Ford [20] prove that the error M2 for the implicit quadrature method behaves as,

$$\max_{j=0,1,\ldots,n} |y(t_j) - y_j| = O(h^{2-q}) \tag{9.9}$$

and the predictor corrector method as

$$\max_{j=0,1,\ldots,n} |y(t_j) - y_j| = O(h^{1+q}), \tag{9.10}$$

where q is the greatest common divisor of the equations constituent derivatives.

9.1.1 Distributed Diethelm (Implicit Quadrature) Method

We now show experimentally that these two errors behave as indicated. A procedure is introduced, which, when incorporated with the graphical technique introduced in section 6.3, enables determination of numerically efficient step sizes for both the quadrature formula and the Diethelm and Ford method.

Table 9.1 shows the time and KFlop count with $N = 2, 4, 8, 16, 32$ and 64 nodes, for the test equation 9.1. For $N = 2, 4$ and 8 nodes the KFlop count is roughly the same for any given step size. This is because the method for $N = 2, 4$ and 8 nodes produces a system of equations of the same size for stage 1 of the numerical scheme. As N increases above 8 the work done increases exponentially. We have

used a sample point of $y = 0.5$ so as to avoid the singularity at $y = 1$. From experimentation we can be reasonably sure that the stated behaviour would be similar for other sample points. In table 9.3 we illustrate this for sample points of $y(0.25)$ and $y(0.75)$.

For very small N (2 and 4) it is noticeable that the $EOC \rightarrow 0$ as $h \rightarrow 0$. From experimentation we have seen this process continue as N increases. We do however require a very small step size h to achieve this. This is due to the numerical scheme possessing two sources of error. For a fixed N, the scheme fixes the error $M1$ associated with the discretization of the distributed order. As $h \rightarrow 0$, the solution therefore tends to $y(T) + M1_N$. Where $M1_N$ is the error associated with the discretization of the distributed order with N nodes. We can use this to calculate the experimental order of convergence EOC for the M2 error in a slightly different way than previously (see formula 5.17). We denote EOC_N as the experimental order of convergence of the $M2$ error and calculate using the formula

$$EOC_N = log_2 \left(\frac{\mid y(T) + M1_N - y_n(T) \mid}{\mid y(T) + M1_{2N} - y_n(T) \mid} \right). \tag{9.11}$$

As N increases the error M1 associated with the discretization of the distributed order, becomes small. Now the EOC starts to converge to the convergence order of the underlying multi term equation algorithm, which in this case is the Diethelm method (see section 7.2.1). As $1/N = H \rightarrow 0$ the $EOC \rightarrow 2$. In table 9.3 we show that the above behaviour is consistent for different sample points.

Table 9.2 shows the EOC_N calculated using the formula (9.11) for $N = 2 \rightarrow 4, 4 \rightarrow 8$ and $8 \rightarrow 16$. We can see that as N increases the step size required for a convergence order of $O(h^2)$ decreases.

h	Diethelm's method (N=2)				Diethelm's method (N=4)			
	error y(0.5)	KFlops	Time	EOC	error y(0.5)	KFlops	Time	EOC
1/8	3.14e-03	6.0	0.00		5.62e-03	6.5	0.00	
1/16	1.37e-03	12.0	0.00	1.20	1.13e-03	12.9	0.00	2.31
1/32	2.76e-03	24.9	0.00	-1.01	2.47e-04	26.7	0.00	2.19
1/64	3.17e-03	54.5	0.05	-0.20	6.61e-04	57.9	0.00	-1.42
1/128	3.30e-03	128.1	0.06	-0.06	7.83e-04	135.0	0.05	-0.24
1/256	3.33e-03	331.5	0.05	-0.01	8.18e-04	339.5	0.11	-0.06

h	Diethelm's method (N=8)				Diethelm's method (N=16)			
	error y(0.5)	KFlops	Time	EOC	error y(0.5)	KFlops	Time	EOC
1/8	6.23e-03	7.8	0.00		5.79e-03	37.7	0.00	
1/16	1.76e-03	15.4	0.00	1.82	1.69e-03	74.7	0.00	1.78
1/32	3.77e-04	31.5	0.06	2.22	4.56e-04	147.8	0.00	1.89
1/64	3.67e-05	66.8	0.05	3.36	9.35e-05	299.4	0.05	2.29
1/128	1.58e-04	152.5	0.05	-2.11	1.08e-05	623.4	0.06	3.11
1/256	1.94e-04	367.0	0.11	-0.30	4.04e-05	1368.0	0.17	-1.90
h	Diethelm's method (N=32)				Diethelm's method (N=64)			
	error y(0.5)	KFlops	Time	EOC	error y(0.5)	KFlops	Time	EOC
1/8	5.55e-03	220.2	0.00		5.42e-03	1459.9	0.11	
1/16	1.63e-03	431.0	0.00	1.77	1.59e-03	2835.0	0.11	1.77
1/32	4.60e-04	840.1	0.05	1.83	4.54e-04	5416.0	0.17	1.81
1/64	1.21e-04	1652.3	0.06	1.93	1.26e-04	10462.3	0.33	1.85
1/128	2.46e-05	3268.0	0.16	2.30	3.27e-05	20114.6	0.55	1.95
1/256	2.56e-06	6576.8	0.33	3.26	6.68e-06	39191.6	1.10	2.29

Table 9.1: Behaviour of the distributed Diethelm method

h	Diethelm's method (EOC_N)		
	EOC_2	EOC_4	EOC_8
1/8	-0.84	-0.15	0.11
1/16	0.28	-0.64	0.06
1/32	3.48	-0.61	-0.27
1/64	2.26	4.17	-1.35
1/128	2.08	2.31	3.87
1/256	2.01	2.08	2.26

Table 9.2: EOC_N for the distributed Diethelm method

h	Diethelm's method (N=8)				Diethelm's method (N=16)			
	error y(0.25)	KFlops	Time	EOC	error y(0.25)	KFlops	Time	EOC
1/8	5.25e-03	4.1	0.00		4.97e-03	19.3	0.00	
1/16	1.58e-03	7.8	0.00	1.73	1.53e-03	37.6	0.00	1.70
1/32	3.80e-04	15.4	0.0 0	2.06	4.31e-04	73.2	0.00	1.83
1/64	9.15e-05	31.2	0.00	2.05	9.94e-05	145.5	0.05	2.12
1/128	1.02e-04	66.8	0.00	-0.57	2.02e-06	293.3	0.06	5.62
1/256	1.35e-04	144.9	0.06	-0.40	2.61e-05	608.7	0.10	-3.69
h	Diethelm's method (N=32)				Diethelm's method (N=64)			
	error y(0.25)	KFlops	Time	EOC	error y(0.25)	KFlops	Time	EOC
1/8	4.82e-03	111.0	0.00		4.73e-03	732.1	0.05	
1/16	1.48e-03	215.9	0.00	1.70	1.45e-03	1418.8	0.06	1.71
1/32	4.31e-04	418.6	0.05	1.78	4.25e-04	2705.8	0.11	3.48
1/64	1.18e-04	817.8	0.06	1.87	1.21e-04	5215.0	0.16	1.81
1/128	2.69e-05	1597.7	0.11	2.13	3.24e-05	9985.5	0.27	1.90
1/256	8.41e-07	3139.9	0.17	5.00	7.37e-06	19301.3	0.60	2.14

h	Diethelm's method (N=8)				Diethelm's method (N=16)			
	error y(0.75)	KFlops	Time	EOC	error y(0.75)	KFlops	Time	EOC
1/8	6.76e-03	11.6	0.00		6.17e-03	56.2	0.00	
1/16	1.90e-03	23.3	0.05	1.83	1.78e-03	112.4	0.00	1.79
1/32	4.35e-04	48.8	0.06	2.13	4.81e-04	224.7	0.00	1.89
1/64	9.51e-08	107.2	0.05	12.16	1.05e-04	462.8	0.06	2.20
1/128	1.27e-04	257.7	0.05	-10.38	2.45e-06	991.5	0.17	5.42
1/256	1.63e-04	667.0	0.17	-0.36	3.27e-05	2278.8	0.33	-3.74
h	Diethelm's method (N=32)				Diethelm's method (N=64)			
	error y(0.75)	KFlops	Time	EOC	error y(0.75)	KFlops	Time	EOC
1/8	5.88e-03	329.7	0.06		5.72e-03	2188.3	0.11	
1/16	1.67e-03	647.1	0.06	1.82	1.65e-03	4253.5	0.11	1.79
1/32	4.77e-04	1266.1	0.11	1.81	4.67e-04	8135.6	0.28	1.82
1/64	1.26e-04	2505.4	0.16	1.92	1.29e-04	15746.7	0.44	1.86
1/128	2.70e-05	5013.2	0.27	2.22	3.26e-05	30392.2	0.88	1.98
1/256	5.84e-07	10312.6	0.49	5.53	7.21e-06	59675.9	1.87	2.18

Table 9.3: Behaviour of the distributed Diethelm method for different sample points

h	Adams method (N=2)(M=16)				Adams method (N=4)(M=16)			
	error y(0.5)	KFlops	Time	EOC	error y(0.5)	KFlops	Time	EOC
1/8	2.34e-02	7.7	0.00		3.06e-03	14.5	0.05	
1/16	3.20e-03	17.8	0.05	2.87	4.89e-03	33.8	0.06	-0.68
1/32	1.70e-03	45.8	0.05	0.91	1.28e-03	87.0	0.11	1.93
1/64	2.87e-03	133.0	0.11	-0.76	8.53e-04	252.1	0.22	0.59
1/128	3.20e-03	431.8	0.22	-0.16	8.24e-04	817.4	0.38	0.05
1/256	3.30e-03	1527.0	0.49	-0.04	8.28e-04	2888.0	0.93	-0.07

h	Adams method (N=8)(M=32)				Adams method (N=16)(M=64)			
	error y(0.5)	KFlops	Time	EOC	error y(0.5)	KFlops	Time	EOC
1/8	1.93e-03	28.2	0.05		1.84e-02	96.9	0.11	
1/16	7.02e-04	65.7	0.11	1.46	2.07e-03	231.4	0.22	3.15
1/32	1.14e-04	169.1	0.16	2.62	3.74e-04	612.4	0.44	2.47
1/64	1.17e-04	490.2	0.38	-0.04	5.67e-05	1825.0	0.82	2.72
1/128	1.83e-04	1588.5	0.77	-0.65	2.28e-06	6046.5	1.70	4.64
1/256	2.01e-04	5609.8	1.81	-0.14	2.41e-05	21677.8	3.96	-3.40

h	Adams method (N=32)(M=128)				Adams method (N=64)(M=256)			
	error y(0.5)	KFlops	Time	EOC	error y(0.5)	KFlops	Time	EOC
1/8	3.50e-02	357.1	0.28		6.80e-02	1368.9	0.55	
1/16	3.10e-03	865.5	0.44	3.50	5.00e-03	3343.9	1.04	3.77
1/32	4.90e-04	2328.2	0.88	2.66	6.16e-04	9070.2	2.15	3.02
1/64	1.07e-04	7036.7	1.93	2.20	1.27e-04	27628.3	4.72	2.28
1/128	1.89e-05	23587.1	4.29	2.50	3.03e-05	93166.6	10.76	2.07
1/256	4.37e-06	85220.5	10.10	2.11	5.80e-06	337931.9	28.51	2.39

Table 9.4: Behaviour of the distributed Adams method

9.1.2 Distributed Adams (Predictor Corrector) Method

As an alternative to the distributed implicit quadrature technique [18] we can use the Adams method from section 5.2, to solve the equation (9.7).

Table 9.4 shows the time and KFlop count for $N = 2, 4, 8, 16, 32$ and 64 nodes. For $N = 2, 4$ and 8 nodes, even though the same order of system is required as for the distributed Diethelm method, the time and KFlop count increase with increasing N. This is due to the increase in corrector iterations needed for convergence. From chapter 7 we know that as the order of the system q decreases, more corrector iterations are required for convergence. In this section we apply enough correctors for convergence, in the next section we determine the optimum level of corrector iterations .

h	N=8, M=64		N=16, M=64		N=32, M=128	
	M1+M2 error y(1)	M2 est	M1+M2 error y(1)	M2 est	M1+M2 error y(1)	M2 est
1/800	-1.2449e-04		-3.0429e-05		-6.847e-06	
1/1600	-1.2501e-04	5.2e-07	-3.1072e-05	6.43e-07	-7.566e-06	7.2e-07
1/3200	-1.2514e-04	1.3e-07	-3.1241e-05	1.67e-07	-7.757e-06	1.9e-07
1/6400	-1.2518e-04	4.0e-08	-3.1286e-05	4.50e-08	-7.808e-06	5.1e-08

Table 9.5: Determining error estimates for the distributed Adams method

Determining the Optimum Number of Corrector Iterations

The error M2, associated with the approximation of a multi-term equation using the Adams method, is of the order $O(h^{1+q})$, where q is the greatest common divisor of constituent derivatives. It was shown in chapter 7, that this can be improved to $O(h^2)$ with sufficient corrector iterations M. In that chapter we introduced a graphical technique to determine the most numerically efficient value for M. To apply this technique to distributed order problems we first need to eliminate the error M1 representing the discretization of the distributed order. For a number of sections N, by reducing the error M2 below a certain error tolerance, using sufficiently small step size h and sufficiently large corrector iterations m, we determine the error M1. Firstly, we determine the number of corrector iterations for a given N, required to achieve convergence up to a particular error tolerance $1.0e-08$. We then continually decrease the step size until an error tolerance of $M2 < 1.0e-07$ is achieved. In table 9.5 we determine an estimate of M1 for $N = 8, 16, 32$ of -1.2518e-04, -3.1286e-05 and -7.808e-06.

We can now determine the optimum number of corrector iterations using a variation of the graphical technique introduced in section 6.3. Instead of plotting the $\log_2(error)$ against $\log_2(time)$, we now plot $\log_2(error - M1_N(h))$ against $\log_2(time)$, where $M1_N(h)$ is the M1 error calculated using N sections with a fixed size of h.

In figures 9.1 to 9.3, we determine that the optimum number of corrector iterations for $N = 8, 16, 32$ are $M = 5, 12, 17$. Some of the lines in the figures are quite eratic, however as $h \to 0$, with sufficiently small H the paths stabilize.

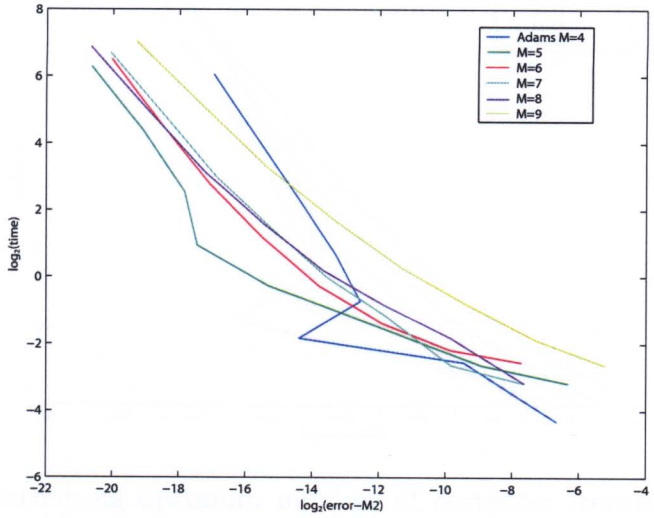

Figure 9.1: Determining optimum number of corrector iterations for $N = 8$ distributed order approximation

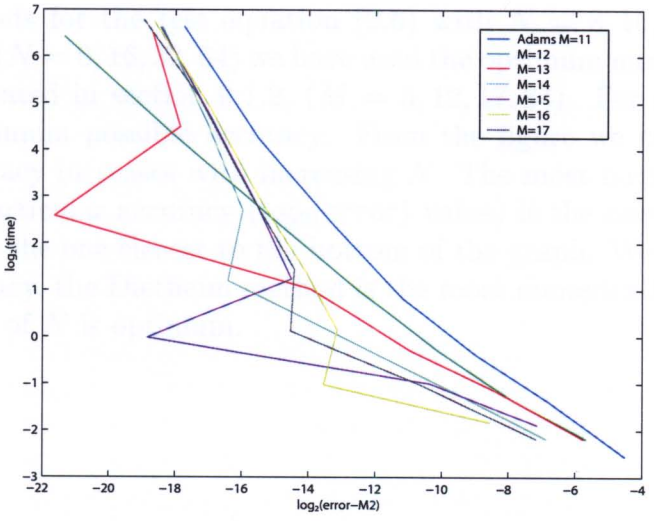

Figure 9.2: Determining optimum number of corrector iterations for $N = 16$ distributed order approximation

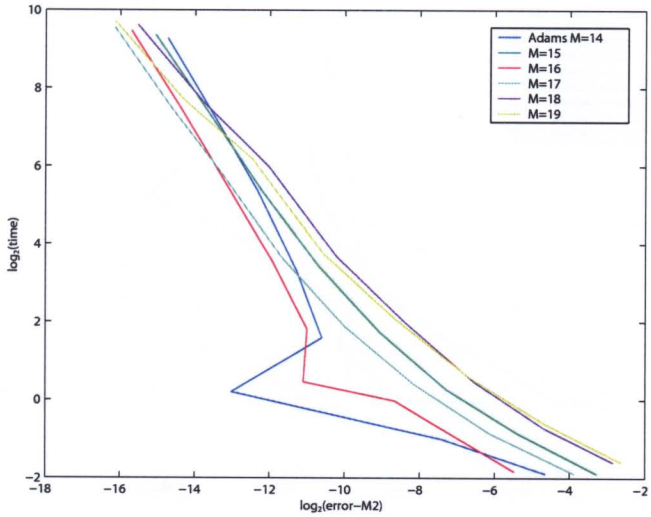

Figure 9.3: Determining optimum number of corrector iterations for $N = 32$ distributed order approximation

9.2 Comparing the Distributed Adams and Diethelm Methods for Numerical Efficiency

In figure 9.4 we compare the numerical efficiency of the distributed Adams and Diethelm methods for the test equation (9.6) with $N = 8, 16, 32, 64$. For the Adams method ($N = 8, 16, 32, 64$) we have used the optimum number of corrector iterations calculated in section 9.1.2, ($M = 5, 12, 17, 34$). For each value of N we have a maximum possible accuracy. From the figure we can see that this maximum accuracy increases with increasing N. The most numerically efficient method for a particular accuracy ($log_2(error)$ value) is the one with the lowest $log_2(time)$, thus the one closest to the bottom of the graph. We can see that for any given accuracy, the Diethelm method is the most numerically efficient and a particular value of N is optimum.

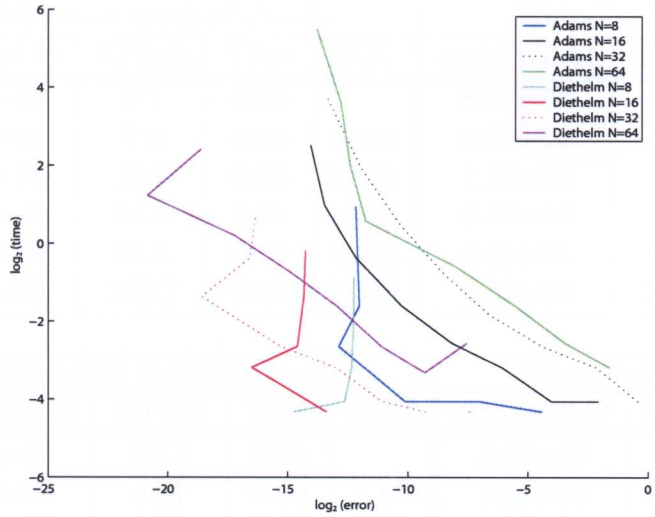

Figure 9.4: Comparison of Adams and Diethelm methods for solving a DODE

9.3 Distributed Midpoint Method

We now explore the possibility of using the midpoint method, as an alternative to the trapezoidal rule, to discretize the distribution integral of the test equation (9.7). The midpoint rule,

$$y_{n+2} = y_n + 2H\phi_{n+1} \tag{9.12}$$

where H is the step size, is convergent of order 2. The composite midpoint rule [59] can be written as

$$Q_{N+1}^{Mp}[\phi] = H \sum_{i=0}^{N-1} \phi(\alpha_i). \tag{9.13}$$

We shall discover however, that due to the process of converting the discretization into a system of equations, the method is more computationally expensive than the trapezoidal method. The method however does have potential value. In chapter 11 we introduce higher order methods for discretizing the distribution integral. These methods give best results when the lower integrand is greater than zero. In the future the distributed midpoint method could possible overcome this limitation.

In table 9.6 we display the error, time and KFlop count for a series of decreasing step size h, for $N = 2, 4, 8, 16, 32, 64$. Note that unlike the Diethelm and Adams methods the midpoint method does not produce the same size system for $N = 2, 4, 8$. For $N = 2$ the midpoint method has differentials of 0.3 and 0.7 thus producing a system of order 0.1. Whereas for $N = 4$ the

h	Midpoint method (N=2)				Midpoint method (N=4)			
	error y(.5)	KFlops	Time	EOC	error y(0.5)	KFlops	Time	EOC
1/8	8.06e-03	3.7	0.00		8.22e-03	1.8	0.00	
1/16	3.60e-03	7.4	0.00	0.736	2.91e-03	3.4	0.00	1.50
1/32	2.22e-03	15.6	0.00	1.125	1.19e-03	7.0	0.00	1.29
1/64	1.81e-03	34.8	0.05	1.196	6.53e-04	16.0	0.00	0.87
1/128	1.69e-03	84.8	0.06	1.408	4.86e-04	40.9	0.00	0.43
1/256	1.65e-03	230.7	0.11	2.315	4.36e-04	118.4	0.11	0.16

h	Midpoint method (N=8)				Midpoint method (N=16)			
	error y(0.5)	KFlops	Time	EOC	error y(0.5)	KFlops	Time	EOC
1/8	5.94e-03	22.3	0.00		5.59e-03	109.5	0.00	
1/16	1.85e-03	44.7	0.00	1.68	1.67e-03	217.1	0.00	1.80
1/32	6.11e-04	90.5	0.00	1.60	4.99e-04	433.5	0.05	1.79
1/64	2.49e-04	189.5	0.06	1.30	1.60e-04	874.6	0.06	1.77
1/128	1.45e-04	407.9	0.05	0.78	6.35e-05	1794.4	0.22	1.68
1/256	1.15e-04	958.7	0.17	0.33	3.64e-05	3815.9	0.33	1.37
h	Midpoint method (N=32)				Midpoint method (N=64)			
	error y(0.5)	KFlops	Time	EOC	error y(0.5)	KFlops	Time	EOC
1/8	5.44e-03	589.0	0.11		5.36e-03		0.22	
1/16	1.60e-03	1162.8	0.11	1.77	1.57e-03		0.33	1.77
1/32	4.64e-04	2261.3	0.11	1.79	4.51e-04		0.66	1.80
1/64	1.36e-04	4458.0	0.28	1.77	1.28e-04		1.26	1.82
1/128	4.25e-05	8818.0	0.49	1.68	3.68e-05		2.42	1.80
1/256	1.64e-05	17948.5	0.99	1.37	1.13e-05		5.33	1.70

Table 9.6: Behaviour of the distributed midpoint method

midpoint method has differentials of $0.2, 0.4, 0.6$ and 0.8 thus producing a system of order 0.2. And for $N = 8$ the midpoint method has differentials of $0.15, 0.25, 0.35, 0.45, 0.55, 0.65, 0.75$ and 0.85 thus producing a system of order 0.05. With $N = 4$ the method is the most numerically efficient so far.

For $N = 8, 16, 32, 64$ we now revert to the pattern of doubling the system size for doubling N, and a similar behaviour to the Adams and Diethelm methods is observed.

In table 9.6 the KFlops column with N=64 is blank, this is due to limitations in the student edition of Matlab.

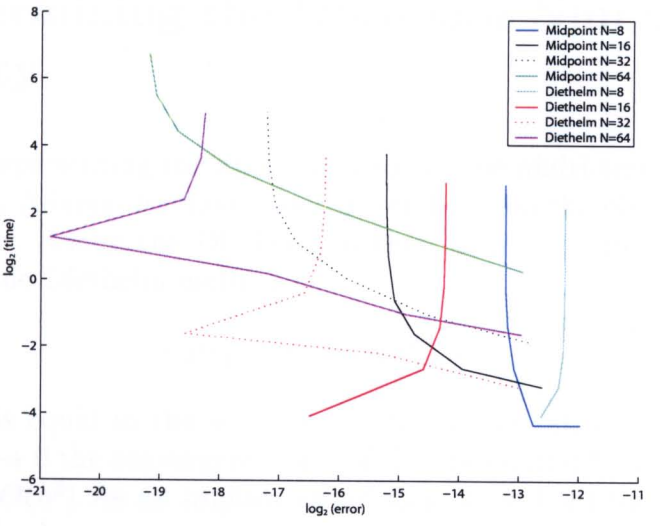

Figure 9.5: Comparison of midpoint and Diethelm methods for solving a DODE

9.4 Comparing the Distributed Midpoint and Diethelm Methods for Numerical Efficiency

In figure 9.5 we compare the numerical efficiency of the distributed midpoint and Diethelm methods for the test equation (9.6) with $N = 8, 16, 32, 64$. For each N the Diethelm method initially outperforms the Midpoint method, then the midpoint method becomes the most numerically efficient. We can see however that the Diethelm method (N_{i+1}) is more numerically efficient than the midpoint method (N_i). This is due to the midpoint method producing a system twice the size of the Diethelm method. This leads to an increase in the work load, but also increases the maximum possible accuracy.

9.5 Determining the Optimum Numerical Efficiency

The error M2, representing the approximation of the multi-term equation using the Diethelm (or Adams) method, has a dependence on the discretization of the distributed order. Using the Diethelm method as an example: from [14], the convergence of the Diethelm method is

$$y(t_j) - y_j = O(h^{2-q}).$$

The value of q is equal to the step size of the discretization of the distributed order H. As $H \to 0$ the convergence rate of the underlying FDE solver increases to $y(t_j) - y_j = O(h^2)$ As an implication of this, if we hold the step size of the underlying FDE solver constant and decrease the step size of the discretization of the distributed order, the error M1 decreases slightly. This decrease becomes increasingly insignificant as the step size of the discretization of the distributed order decreases, and will therefore be ignored in the following procedure.

For a given accuracy ($M1+M2 <$ accuracy required) we would like to determine the optimum ratio of each error component. The process involves several stages, slightly different for M1 or M2. Calculating M2 for the distributed Diethelm method:

1. Keeping the number of sections N constant; determine an approximation for $y(T) + M1_N$, by making the number of steps n large.

2. Using this approximation, obtain an expression for $M2_j$, where $j < n$, and percentage error bound, at a particular step size.

3. By adding and subtracting the appropriate percentage, obtain upper and lower bounds for $M2_j$.

4. Using the convergence order of the Diethelm method, which is shown in [11] to be $O(h^{2-q})$, extrapolate to find the number of steps $n_{required}$ required for $M2 = (AccuracyRequired)/2$. We do this using: $n(i)_{required} = n \times M2(i)_h/(AccuracyRequired)^{(1/(2-q))}$.

5. Examine $abs(n(1) - n(3))/n(2)$; this gives an indication of how reliable the number of steps required is. If $abs(n(1) - n(3))/n(2)$ is relatively large, use of an n value towards the upper bound is desirable; if small, use of the middle bound is recommended.

Calculating M1 for the distributed Diethelm method:

1. Keeping the step size n constant; determine an approximation for $y(T) + M2_n$, by making the number of sections N large.

2. Using this approximation, obtain an expression for $M1_i$, where $i < N$, and percentage error bound, at a particular section size $1/i$.

3. By adding and subtracting the appropriate percentage, obtain upper and lower bounds for $M1_i$.

4. Using the convergence order of the trapezoidal method , which is shown in [59] to be $O(h^2)$, extrapolate to find the number of sections, $N_{required}$, required for $M1 = (AccuracyRequired)/2$. We do this using: $N(i)_{required} = N \times M1(i)_h/(AccuracyRequired)^2$.

5. Examine $abs(N(1) - N(3))/N(2)$; this gives an indication of how reliable the number of steps required is. If $abs(N(1) - N(3))/N(2)$ is relatively large, use of an N value towards the upper bound is desirable; if small, use of the middle bound is recommended.

It is now a simple matter of executing the method with the desired integers n and N.

A similar approach can be used for any other applicable method.

9.6 Conclusions

In this chapter we have introduced the reader to the concept of distributed order differential equations (DODEs). We have described the only numerical scheme in the current literature able to solve DODEs (Diethelm and Ford [14]) and explain how it can be used in conjunction with either the Diethelm or Adams multi-term methods (see chapter 7).

We have shown how the Diethelm and Ford scheme introduces two sources of error, the first from the discretization of the distributed order (M1), and the second from the approximation of the multi-term equation (M2). A new version of the EOC calculation was introduced (9.11) and we show how the convergence order of each error source could be tested.

The distributed Diethelm method (DDM) was compared with the distributed Adams method (DAM) and we discovered that the DDM was always the most numerically efficient.

We then introduced an alternative way to discretize the distribution integral based upon the midpoint rule. This was compared to the DDM and is shown only to be numerically more efficient in limited special cases.

A procedure was introduced, suitable for any distributed method, for obtaining optimum numerical efficiency.

In the next chapter we show how a Richardson extrapolation can be applied to help reduce the number of nodes required for any given accuracy, thus improving computational efficiency.

Chapter 10

Distributed Richardson Extrapolation

Richardson extrapolation is an 'old' technique for speeding up the numerical solution of algorithms, by estimating the local truncation error.

For a fixed value T, we get a sequence of solutions $x_i(T)$, by successively halving the step size of a numerical method. If $x_i(T)$ possesses an asymptotic error expansion (AEE), it is possible to extrapolate to produce a more accurate answer.

In [25], Diethelm and Walz prove the existence of an AEE for a single-term equation approximated using the implicit quadrature [11] algorithm. Due to theorem 7.2.1, which shows how a multi-term equation can be expressed as a single-term equation in vector format, the generalization to multi-term equations is immediate.

In this chapter we first explain the extrapolation algorithm, then show how to use this to reduce the error M1 connected with discretising the distribution integral. Showing the existence of an AEE then proves the validity of this technique.

10.1 Richardson Extrapolation

In this section we follow the approach given by Lambert [41]. Suppose we have a numerical method of order p, and obtain a solution y_{n+1} at t_{n+1}. The truncation error Tr_{n+1} can be written as

$$Tr_{n+1} = y(t_{n+1}) - y_{n+1} = \psi(y(t_n))h^{p+1} + O(h^{p+2}) \tag{10.1}$$

where y_{n+1} indicates the value for y at t_{n+1} under the localizing assumption, and $\psi(y(t_n))$ is a function of the elementary differentials of order $p+1$ evaluated at $y(t_n)$.

We now compute a second numerical solution z_{n+1}, but now use a step length of $2h$, starting at t_{n-1}. Expanding $y(t_{n-1})$ about t_n, we can write

$$\begin{aligned} y(t_{n+1}) - z_{n+1} &= \psi(y(t_{n-1}))(2h)^{p+1} + O(h^{p+2}) \\ &= \psi(y(t_n))(2h)^{p+1} + O(h^{p+2}). \end{aligned} \tag{10.2}$$

Subtracting (10.1) from (10.2) gives

$$(2^{p+1} - 1)h^{p+1}\psi[y(t_n)] = y_{n+1} - z_{n+1} + O(h^{p+2}).$$

Thus we have the principal local truncation error,

$$(y_{n+1} - z_{n+1})/(2^{p+1} - 1).$$

10.2 Richardson Extrapolation of the Order Distribution

The Richardson extrapolation can be used to reduce the error associated with either the discretization of the distributed order (M1), or the error from the solution of the FDE (M2).

In [62], A. C. Simpson gives an illustration of the improved performance using the Richardson extrapolation with the implicit quadrature [11] method for the solution of single-term FDEs. We shall not repeat the findings here, instead we concentrate on using the Richardson extrapolation to reduce the error (M1), of a solution to a DODE, using the distributed Diethelm method 9.1.1.

To apply the Richardson extrapolation to the distributed order problem we need to hold the step size of the underlying FDE solver constant. We then do the following:

1. Determine the maximum order of the system q and set $N = (1 - a)/q$, where a is the lower integrand of the distribution integral.
2. Compute the solution $y_{N_1}(T)$ for the equation with $N_1 = N$.
3. Double the number of terms in the distribution approximation.
4. Compute the solution $y_{N_2}(T)$ for the equation with $N_2 = N/2$.
5. Continue as above and compute $y_{N_3}(T)$,$y_{N_4}(T)$,$y_{N_5}(T)$.

N	time	error1	error2	error3	error4	error5	error6
8	4.12	-5.68e-05					
			8.18e-06				
16	7.36	-1.43e-05		-1.16e-06			
			2.04e-06		1.64e-07		
32	13.46	-3.56e-06		-2.89e-07		-2.12e-08	
			5.07e-07		3.91e-08		8.99e-10
64	31.86	-8.89e-07		-7.03e-08		-3.44e-09	
			1.25e-07		7.90e-09		
128	104.85	-2.20e-07		-1.57e-08			
			2.93e-08				
256	596.49	-5.32e-08					

Table 10.1: Richardson extrapolation of the distribution integral error

6. Extrapolate using the technique introduced in section 10.1.

We use the test equation,

$$\int_{0.1}^{0.9} \Gamma(3-\alpha) D^{\alpha} y(t) d\alpha = 2\frac{t^{1.9} - t^{1.1}}{\ln t}, \text{ y(0)=0,} \tag{10.3}$$

which has an exact solution of $y = t^2$.

Table 10.1 illustrates the reduction of error at $t = 0.1$ with a step size of $1/20000$. We use a small step size so as to ensure the M2 error does not affect the convergence order of the M1 error.

As the value N increases the run time required increases exponentially. The combined time up to $N = 128$ is 161.65 seconds and it is possible to extrapolate down to an error of 2.12e-08. This is a big improvement on the direct use of $N = 256$.

10.3 Asymptotic Error Expansion of the Distributed Diethelm Method

When the trapezoidal rule is applied to discretize the integral of a distributed order equation the following arises: The total error consists of 2 parts, M1 and M2, from the trapezoidal rule and the underlying FDE method respectively.

Let a series of solutions y_{N_i} of the DODE equation (10.3) be given. The error expansion of the trapezoidal rule in stage 1 of the algorithm possesses an

asymptotic error expansion of the form

$$y_N(T) = y(T) + \psi(y(T))H^3 + O(H^4) \tag{10.4}$$

where N is the number of steps in the approximation of the trapezoidal rule.

Diethelm and Walz [25] prove that the error of Diethelm's [11] underlying FDE method possesses an asymptotic error expansion of the form,

$$y_n = y(t_n) + \sum_{\mu=2}^{M_1} c_\mu n^{q-\mu} + \sum_{\mu=1}^{M_2} c_\mu n^{-2\mu} + O(n^{-M_3})$$

for $h \to 0$, $M_3 = \min\{q - M_1, 2M_2\}$.

For the distributed order problem $q = H$, where H is the step size in the distribution discretization. For any given solution $y(T) = y(t_n)$ and $y_N(T) = y_n$. To get the total local error we add the constituent error parts, therefore

$$y_n - y(t_n) = [M1] + [M2]$$

$$y_n = y(t_n) + \psi(y(t_n))H^3 + \sum_{\mu=2}^{M_1} c_\mu n^{q-\mu} + \sum_{\mu=1}^{M_2} c_\mu n^{-2\mu} + O(n^{-M_3}) + O(H^4)$$

Note: we need the error expansion as $n \to \infty$, which is asymptotic.

10.4 Conclusions

In this chapter we have introduced a Richardson based extrapolation algorithm, which increases the accuracy of the Diethelm and Ford [14] distributed method. We prove that the Diethelm and Ford algorithm, used in conjunction with the trapezoidal quadrature fomula (9.4) and the implicit quadrature method from section 5.1, possesses an asymptotic error expansion (AEE).

We demonstrated that for any required accuracy the extrapolation significantly improves numerical efficiency and that the difference in performance increases as desired accuracy increases.

In the next chapter we introduce a new high order method for the discretization of the distribution integral in the DODE (9.2).

Chapter 11

Distributed Trapezoidal Rules

11.1 Motivation

In chapter 9 we gave an exposition of Diethelm and Ford's scheme [14] for solving distributed order differential equations (DODEs) of the form,

$$\int_a^b A(\alpha) D^{m+\alpha} y(t) d\alpha = g(t). \tag{11.1}$$

Diethelm and Ford use the trapezoidal quadrature formula (9.4), to discretize the integral in equation (11.1). It is well established that the trapezoidal quadrature formula produces an error approximation of order $O(h^2)$ (see formula (9.8)).

In section 9.1.1 we demonstrated that the error $M2$, associated with discretizing the distribution integral in (11.1) with the trapezoidal quadrature formula, is dominant when the number of steps N is relatively small. As N increases the number of terms created in stage one of the algorithm increases, this leads to increased computational complexity. To reduce run times it would be desirable to limit the size of the required system.

In this chapter we apply extended trapezoidal rules to the distributed order differential equation (11.1). This increases the order of convergence, thus reducing the size of the required system.

11.2 Extended Trapezoidal Rules

In [3] Brugnano and Trigiante show how extended trapezoidal rules (ETRs) can be used to solve boundary value problems for ordinary differential equations (ODEs).

ETRs originate from generalized Adams methods (GAMs) which have an odd number of steps. GAMs are defined as having a characteristic polynomial of the form

$$\rho(z) = z^{j-1}(z-1), \quad j = 1, \ldots, k, \tag{11.2}$$

where $j = \frac{k+1}{2}$ for odd k, and $j = \frac{k}{2}$ for even k.

Thus ETRs are defined by,

$$y_{n+v} - y_{n+v-1} = H \sum_{i=0}^{2v-1} \beta_i f_{n+i}, \tag{11.3}$$

where $H = 1/N$ is the step length.

The weights β_i are obtained by solving the linear system,

$$W_0^{(k)} \begin{pmatrix} \beta_o \\ \beta_1 \\ \vdots \\ \beta_k \end{pmatrix} = \begin{pmatrix} 1 \\ \frac{j^2-(j-1)^2}{2} \\ \vdots \\ \frac{j^{k+1}-(j-1)^{k+1}}{k+1} \end{pmatrix}, \tag{11.4}$$

with $j = v$, where $W_0^{(k)}$ is the Vandermonde matrix (see [3]).

Brugnano and Trigiante show that all the methods are $O_{v,v-1}$-stable by construction, can be shown to be $A_{v,v-1}$-stable and the boundary loci are regular Jordan curves.

11.2.1 Trapezoidal Rule

For $v = 1$ we obtain the trapezoidal rule,

$$y_{n+1} - y_n = \frac{H}{2}(f_{n+1} + f_n), \tag{11.5}$$

where $H = 1/N$. Summing for a fixed N gives

$$\sum_{n=0}^{N}(y_{n+1} - y_n) = \frac{H}{2}f_0 + H(f_1 + \cdots + f_n) + \frac{H}{2}f_{n+1}. \tag{11.6}$$

11.2.2 Fourth Order ETR

For $v = 2$, we obtain a fourth order method

$$y_{n+1} - y_n = \frac{H}{24}\left(-f_{n+2} + 13f_{n+1} + 13f_n - f_{n-1}\right), \quad n = 1, \ldots, N-2. \quad (11.7)$$

Summing for a fixed N gives

$$\begin{aligned}\sum_{n=0}^{N}(y_{n+1} - y_n) &= \frac{H}{24}\left(-f_{-1} + 12f_0 + 25f_1 + 24(f_2 + \cdots + f_{n-1})\right. \\ &\quad \left. +25f_n + 12f_{n+1} - f_{n+2}\right). \quad (11.8)\end{aligned}$$

11.2.3 Sixth Order ETR

For $v = 3$ we obtain a sixth order method

$$y_{n+1} - y_n = \frac{H}{1440}\left(11f_{n+3} - 93f_{n+2} + 802f_{n+1} + 802f_n - 93f_{n-1} + 11f_{n-2}\right), \quad (11.9)$$

for $n = 2, \ldots, N-2$.

Summing for a fixed N gives

$$\begin{aligned}\sum_{n=0}^{N}(y_{n+1} - y_n) &= \frac{H}{1440}\left(11f_{-2} - 82f_{-1} + 720f_0 + 1522f_1 + 1429f_2\right. \\ &\quad +1440(f_3 + \cdots + f_{n-2}) \\ &\quad \left. +1429f_{n-1} + 1522f_n + 720f_{n+1} - 82f_{n+2} + 11f_{n+3}.\right)\end{aligned}$$

11.3 Distributed ETR Method

We can see from formulae (11.7) and (11.9) that for any given step, previous and future values of the function are required. When using ETRs for solving ODEs, the user requires a starting and an ending procedure to obtain these values. For example, if we consider a problem where the initial condition $y(0) = y_0$ is known, using the fourth order ETR (11.7) to calculate the integral between y_0 and y_1, the user would require the function value at y_{-1} and y_2. We can get round this by using the standard trapezoidal rule for the first and last steps.

As shown in chapter 9 the DODE scheme introduced by Diethelm and Ford in [14] is a two stage process. The first stage, the approximation of the distribution

integral in equation (11.1) by a multi-term equation, must be completed before the second stage commences.

Therefore, for stage 1 of Diethelm and Ford's DODEs scheme, the starting and ending procedure can be applied by adding the trapezoidal approximation between y_0 and y_1 and the trapezoidal approximation between y_n and y_{n+1}, to the fourth order approximation (11.8) from y_1 to y_n.

This gives us a new method,

$$\begin{aligned}\sum_{n=0}^{N}(y_{n+1}-y_n) &= \frac{H}{24}(11f_0+24f_1+25f_2+24(f_3+\cdots+f_{n-2}) \\ &\quad +25f_{n-1}+24f_n+11f_{n+1}) \end{aligned} \tag{11.10}$$

which possesses a convergence order $O(H^p)$ where $2 < p < 4$. We shall refer to the method as the 'fractional ETR'.

The 'fractional ETR' can be used to obtain an approximation of the integral in the test equation,

$$\int_{0.1}^{0.9}\Gamma(3-\alpha)D^{\alpha}y(t)d\alpha = 2\frac{t^{1.9}-t^{1.1}}{\ln t},\ \text{y(0)=0}, \tag{11.11}$$

which has an exact solution of $y = t^2$.

For example, discretizing (11.11) using (11.10), with $N = 8$, we get the multi-term equation,

$$\begin{aligned}&1.1\Gamma(2.9)D^{0.1}y(t)+2.4\Gamma(2.8)D^{0.2}y(t)+2.5\Gamma(2.7)D^{0.3}y(t) \\ &+2.4\Gamma(2.6)D^{0.4}y(t)+2.4\Gamma(2.5)D^{0.5}y(t)+2.4\Gamma(2.4)D^{0.6}y(t) \\ &+2.5\Gamma(2.3)D^{0.7}y(t)+2.4\Gamma(2.2)D^{0.8}y(t)+1.1\Gamma(2.1)D^{0.9}y(t)\end{aligned} = 2\frac{t^{1.9}-t^{1.1}}{\ln t}, \tag{11.12}$$

together with the initial condition $y(0) = 0$.

In table 11.1 we show the error of the distributed trapezoidal method (DTM) and the 'fractional ETR' for a series of decreasing step sizes and increasing number of nodes.

For any given N, there is an error M1 associated with the discretization of the distribution integral in equation (11.11). As we decrease the step size h of the underlying FDE solver the error $y(t_n) - y_n$ converges to M1. Table 11.1 shows how the new 'fractional ETR' lowers the value of M1 for any given value of N.

In table 11.2 we use the EOC_N formula (9.11) to experimentally determine the order of convergence for the M1 error. For the trapezoidal rule, as expected, $EOC_N \to 2$, as $h \to 0$. The EOC_N for the 'fractional ETR' appears to be converging to 3 as $h \to 0$. As N increases the step size required for this to occur is reduced. Note a negative EOC represents an order of convergence which is divergent and that reducing the step size h only reduces the $M1$ error (see section 9.1).

Approx M1 Error (abs) of the distributed trapezoidal method at t=0.5							
steps	number of sections (N)						
	8	16	32	64	128	256	512
100	1.31e-04	1.26e-05	4.62e-05	5.35e-05	5.48e-05	5.48e-05	5.47e-05
200	1.86e-04	3.49e-05	3.49e-06	1.25e-05	1.45e-05	1.50e-05	1.50e-05
400	2.01e-04	4.69e-05	8.44e-06	1.06e-06	3.39e-06	3.94e-06	4.07e-06
800	2.06e-04	5.05e-05	1.17e-05	2.07e-06	3.30e-07	9.22e-07	1.07e-06
1600	2.07e-04	5.15e-05	1.26e-05	2.93e-06	5.03e-07	1.01e-07	2.50e-07
3200	2.08e-04	5.18e-05	1.29e-05	3.16e-06	7.28e-07	1.21e-07	

Approx M1 Error (abs) of the 'fractional ETR' at t=0.5						
steps	number of sections (N)					
	8	16	32	64	128	256
100	2.24e-05	5.76e-05	5.84e-05	5.67e-05	5.56e-05	5.50e-05
200	3.25e-05	1.12e-05	1.56e-05	1.56e-05	1.53e-05	1.52e-05
400	4.83e-05	1.91e-06	3.66e-06	4.20e-06	4.19e-06	4.14e-06
800	5.29e-05	5.56e-06	3.62e-07	1.06e-06	1.13e-06	1.12e-06
1600	5.41e-05	6.58e-06	5.42e-07	2.07e-07	2.94e-07	3.02e-07
3200	5.45e-05	6.85e-06	7.88e-07	2.51e-08	6.88e-08	7.96e-08

Table 11.1: Comparing the error of the DTR and the fractional ETR methods

	Trapezoidal rule(EOC_N)		Fractional ETR(EOC_N)	
steps	EOC_8	EOC_{16}	EOC_8	EOC_{16}
100	3.38	-1.87	-1.36	-0.02
200	2.41	3.32	1.54	-0.48
400	2.10	2.47	4.66	-0.94
800	2.03	2.11	3.25	3.94
1600	2.01	2.03	3.04	3.60
3200	2.01	2.01	2.99	3.12

Table 11.2: Comparing EOC_N for the DTM and the fractional ETR methods

11.3.1 Convergence of the Fractional ETR

We now give local and global error bounds for the fractional ETR method.

Theorem 11.3.1 *The M1 error, representing the error from the discretization of the distribution integral in (11.1) using the fractional ETR (11.10), is bounded by,*

$$E_F = |y(t_j) - y_j| \leq \mu H^3, \quad j = 0, 1, \ldots, n,$$

where μ is a constant depending upon y.

Corollary 11.3.2 *The M1 error, representing the error from the discretization of the distribution integral in equation (11.1) using the fractional ETR (11.10), has a global estimate,*

$$\max_{j=0,1,\ldots,n} |y(t_j) - y_j| = O(H^3).$$

Proof of theorem (11.3.1) The local error of the trapezoidal rule is well known [3] to behave as

$$E_t = |y(t_j) - y_j| \leq \gamma H^2,$$

where γ is a constant depending upon y.

By definition the local error of the fourth order ETR can be expressed in a similar manner

$$E_{4th} = |y(t_j) - y_j| \leq \beta H^4,$$

where β is a constant depending upon y.

The fractional ETR method uses the trapezoidal rule as a starting and ending procedure, therefore intervals $a \rightarrow a + 1/n$ and $(b - 1/n) \rightarrow b$ have an error bounded by E_t, thus to obtain the error bound of the fractional ETR we add the error bound of the trapezoidal rule over the stated intervals to the error bound of the fourth order ETR, which gives

$$E_F \leq 2H(\gamma H^2) + \beta H^4$$

$$E_F \leq 2\gamma H^3 + \beta H^4.$$

As $H \rightarrow 0$, $E_f = O(H^3)$. This concludes our proof.

11.4 Fourth and Sixth Order Distributed ETR Implementation

In ODEs, due to dependence on past and future values of the function $f(t)$, fourth and sixth order ETRs require starting and ending procedures. In chapter 9 we explain how Diethelm and Ford's [14] scheme for solving DODEs involves two stages, the approximation of the distribution integral with a multi-term equation, and the solution of this equation with a multi-term FDE solver. The fourth and sixth order ETRs can give a direct expression for the approximation of the distribution integral in the form of a multi-term equation. The fourth and sixth order ETRs require 'past' function values f_{-1} and f_{-1}, f_{-2} respectively; these can be represented by 'past' values of the derivative. As a derivative is not necessarily an inverse of an integral, these 'past' derivatives must be positive. For a subset of equations of the type (11.1), when the lower integrand $a > 0$, we can therefore use the fourth and sixth order ETRs for discretizing the distribution integral.

For example, the integral in equation (11.11), discretized using the fourth order ETR (11.8) with $N = 8$ gives,

$$\begin{array}{l} -0.1\Gamma(3.0)\tilde{y}(t) + 1.2\Gamma(2.9)D^{0.1}\tilde{y}(t) + 2.5\Gamma(2.8)D^{0.2}\tilde{y}(t) \\ +2.4\left(\Gamma(3-3H)D^{3H}\tilde{y}(t) + \cdots + \Gamma(3-7H)D^{7H}\tilde{y}(t)\right) \\ +2.5\Gamma(2.2)D^{0.8}\tilde{y}(t) + 1.2\Gamma(2.1)D^{0.9}\tilde{y}(t) - 0.1\Gamma(2.0)D\tilde{y}(t) \end{array} = 2\frac{t^{1.9} - t^{1.1}}{\ln t}, \tag{11.13}$$

together with the initial condition $\tilde{y}(0) = 0$. We can now use the Diethelm method to produce an approximation for equation (11.13).

If we do the same using the sixth order ETR with $N = 8$, the first term of the multi-term equation would have a derivative $D^{-0.1}y(t)$. For the reasons stated above negative derivatives are not permitted. If we increase the number of nodes so $N = 16$, the values f_{-1}, f_{-2} produce derivatives $D^{0}y(t)$ and $D^{0.05}y(t)$.

Due to the dense nature of real numbers, given a value of N sufficiently large, this technique can be used for all equations of the type (11.1), when the lower integrand $a > 0$.

We again use the DODE test equation (11.11). In table 11.3 we show the error of the fourth order distributed ETR and the sixth order distributed ETR, for a series of decreasing step sizes and increasing number of nodes for the test equation (11.11). For the reasons given previously the sixth order distributed ETR cannot be used for an $N = 8$ discretization.

As N increases, the error associated with the dicretization of the distribution integral, M1, becomes small. The error M2, associated with the underlying FDE

Approx M1 Error (abs) of fourth order trapezoidal method						
steps	number of sections (N)					
	8	16	32	64	128	256
100	7.77e-05	6.46e-05	5.92e-05	5.67e-05	5.56e-05	5.50e-05
200	2.28e-05	1.82e-05	1.65e-05	1.57e-05	1.53e-05	1.52e-05
400	7.00e-06	5.09e-06	4.54e-06	4.31e-06	4.20e-06	4.15e-06
800	2.49e-06	1.44e-06	1.24e-06	1.17e-06	1.14e-06	1.13e-06
1600	1.21e-06	4.27e-07	3.40e-07	3.18e-07	3.08e-07	3.03e-07
3200	8.56e-07	1.49e-07	9.40e-08	8.56e-08	8.27e-08	8.14e-08

Approx M1 Error (abs) of sixth order trapezoidal method						
steps	number of sections (N)					
	8	16	32	64	128	256
100		6.46e-05	5.92e-05	5.68e-05	5.56e-05	5.50e-05
200		1.81e-05	1.65e-05	1.57e-05	1.53e-05	1.52e-05
400		5.04e-06	4.54e-06	4.31e-06	4.20e-06	4.15e-06
800		1.39e-06	1.24e-06	1.17e-06	1.14e-06	1.13e-06
1600		3.82e-07	3.37e-07	3.17e-07	3.08e-07	3.03e-07
3200		1.04e-07	9.12e-08	8.54e-08	8.27e-08	8.14e-08

Table 11.3: Comparing the errors of the fourth and sixth order distributed ETR methods for the test equation 11.11

solver, therefore becomes dominant. The fourth and sixth order distributed ETRs therefore appear to have the same performance. The order of convergence of the M2 error for both methods is $O(h^{2-H})$, where H is the step length between successive nodes. As N increases, the order of convergence tends to 2.

Figure 11.1 shows how the absolute error of each higher order method varies with N, for a fixed step size $h = 1/3200$. Each method converges to the same M1 error value. The fourth and sixth order distributed ETRs converge to the M1 error fastest. For this particular test equation, the fractional ETR's two constituent errors M1 and M2 possess different signs. Therefore the two errors cancel out at certain values of N and the method over performs. For N up to 32, we can see that the performance of the 'fractional ETR' is between that of the trapezoidal rule and the fourth order ETR.

11.4.1 Numerical Cost Comparison

Table 11.4 shows that the work load for the higher order methods is of the same order as for the trapezoidal method.

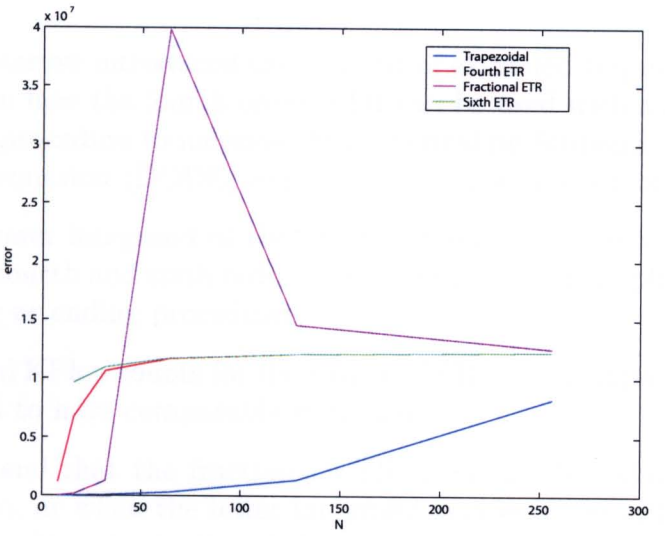

Figure 11.1: Comparison of higher order methods

	Trapezoidal		Fourth ETR		Fractional ETR		Sixth ETR	
Steps	KFlops	Time	KFlops	Time	KFlops	Time		
100	483.9	0.06	564.9	0.06	474.3	0.06	668.2	0.06
200	1040.2	0.11	1226.2	0.11	1040.2	0.11	1437.7	0.11
400	2403.3	0.28	2784.8	0.28	2403.3	0.27	3216.5	0.28
800	6285.6	0.55	7128.5	0.66	6285.6	0.60	8071.8	0.66
1600	18275.8	1.54	20238.3	1.65	18275.8	1.53	22398.4	1.70

Table 11.4: Comparing work done for the distributed ETR methods

11.5 Conclusions

In this chapter we introduced the concept of extended trapezoidal rules (ETRs). It was shown how the fourth order ETR can be used with a starting and ending trapezoidal procedure to increase the numerical performance of distributed order differential equation (DODE) algorithms for all equations of the type (11.1).

When the lower integrand of the order distribution is greater than zero, it was shown that fourth and sixth order ETR methods can be applied directly, without any starting or ending procedures.

The time and KFlop counts for the various ETRs were compared, and all methods where found to have comparable workloads.

We recommend that the fractional ETR be used when the lower integrand of (11.1) is zero, or when the lower integrand is close to zero, forcing the required node number N to be high and thus computationally expensive for the fourth and sixth order methods. In all other equations we recommend the sixth order distributed ETR, however the fourth order method will usually suffice.

In the next chapter we show the schematics of a new automated FDE solver for the solution of FDEs and DODEs.

Chapter 12

Fractional Differential Equation Solver

A fractional differential equation (FDE) solver will be available for download from the University of Chester mathematics website at

`www.chester.ac.uk\mathematics`

The solver is capable of obtaining solutions to single and multi-term FDEs, and distributed order differential equations (DODEs). The numerical efficiency of any appropriate method is determined and approximate run times for a given error tolerance are displayed. The user can then execute the most efficient method.

The program has been produced in a modular format, so additional methods or more efficient code can be substituted seamlessly.

In this chapter we show schematics of the FDE solver and explain how the modular format can benefit future maintainability. We show how new methods can be inserted with minimum effort.

Specific programming issues are addressed.

12.1 FDE Solver Schematics

Figure 12.1 shows a schematic of how the graphical user interfaces (GUI), m-files and output/input files are linked. The interfaces of the m-files and data structures of the GUIs are shown. The direction of arrows shows reliance of one component upon another. For example, 'SET_VARIABLES_DISTRIBUTED' only

functions in conjunction with 'FDE_GUI' and 'DISTRIBUTED_SOLVER', however 'DISTRIBUTED_SOLVER' does not require 'SET_VARIABLES_DISTRIBUTED' to operate, just the correct input data. The indexing on 'DIS_METHODS' and 'MUL_METHODS' represents calls to various different methods; details of the structure of these methods is given in figures 12.2 to 12.7.

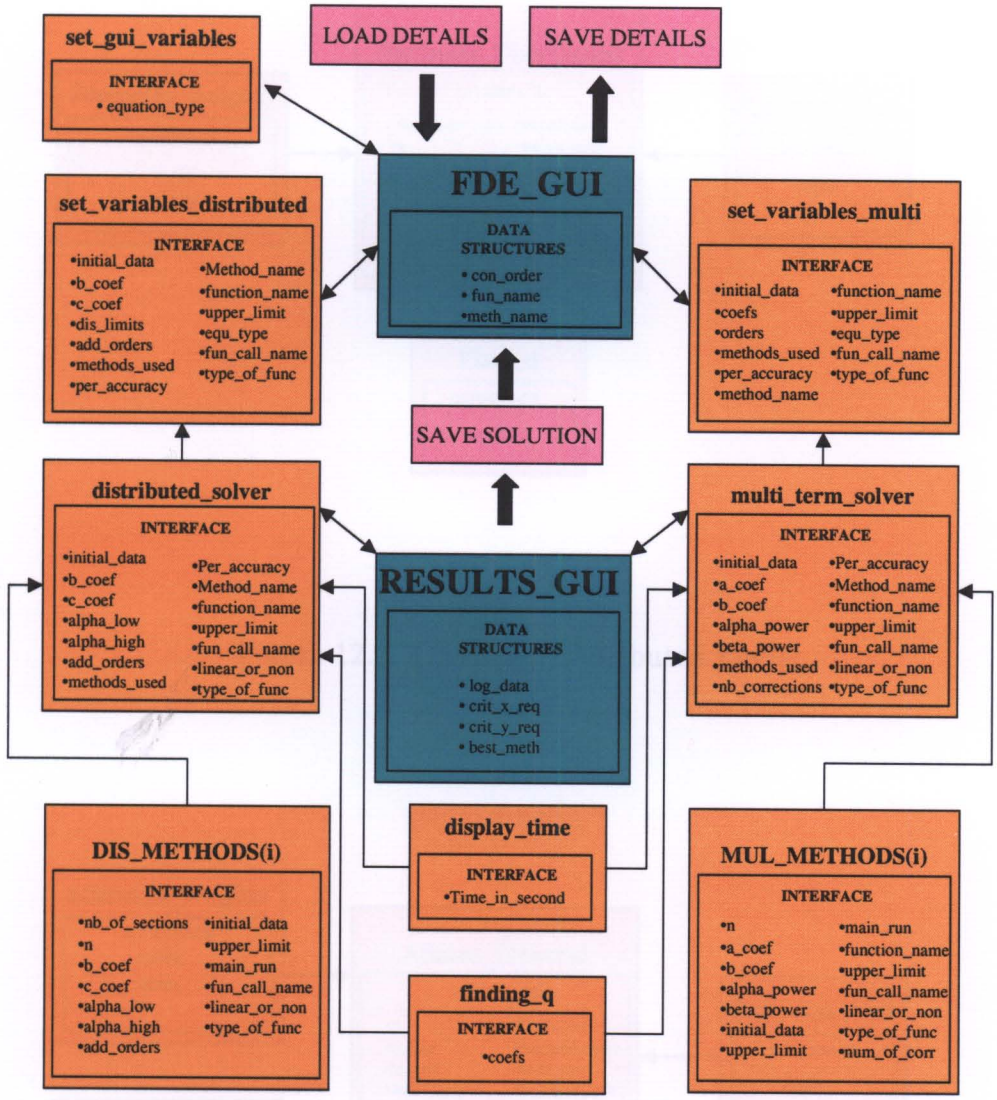

Figure 12.1: Program schematic

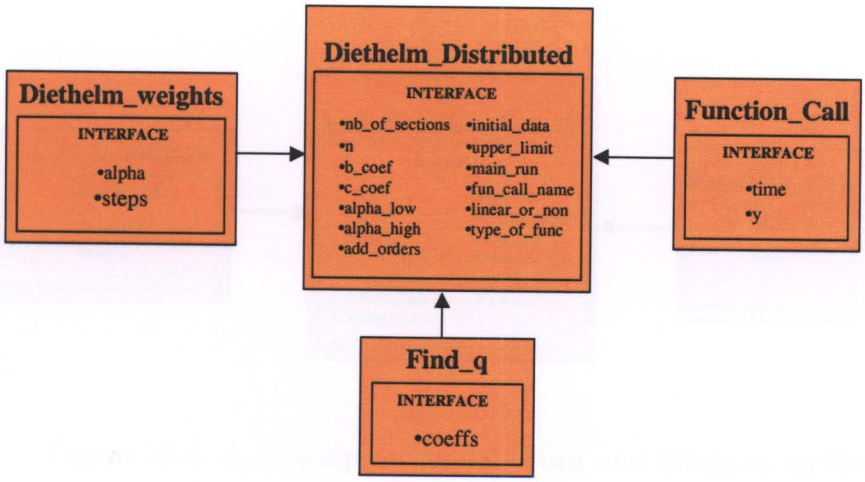

Figure 12.2: Diethelm_Distributed model

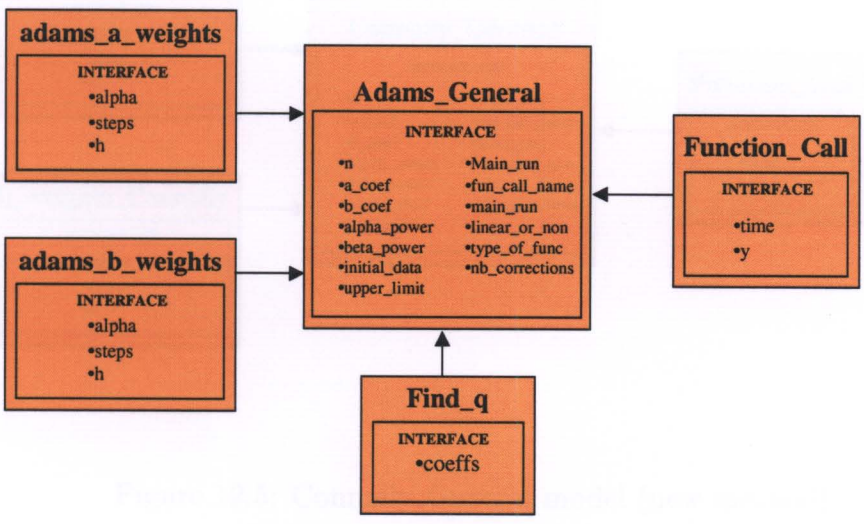

Figure 12.3: Adams_General model

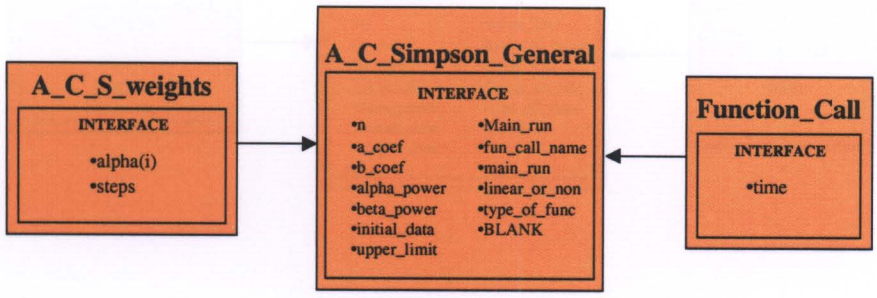

Figure 12.4: A_C_Simpson model (Ford and Simpson method)

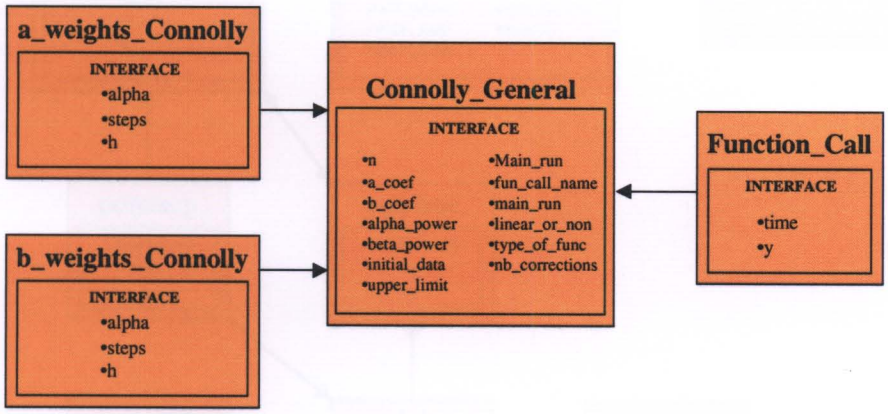

Figure 12.5: Connolly_General model (new method)

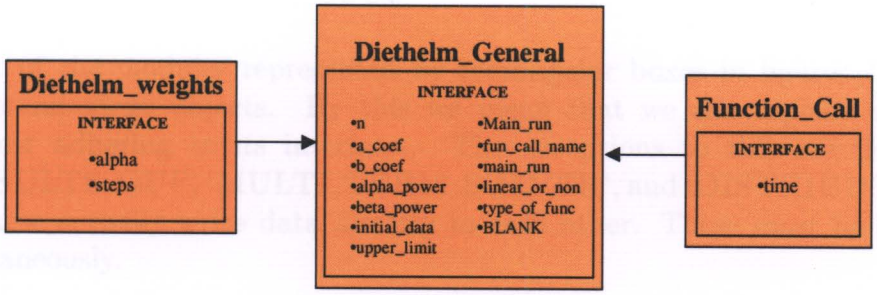

Figure 12.6: Diethelm_General model

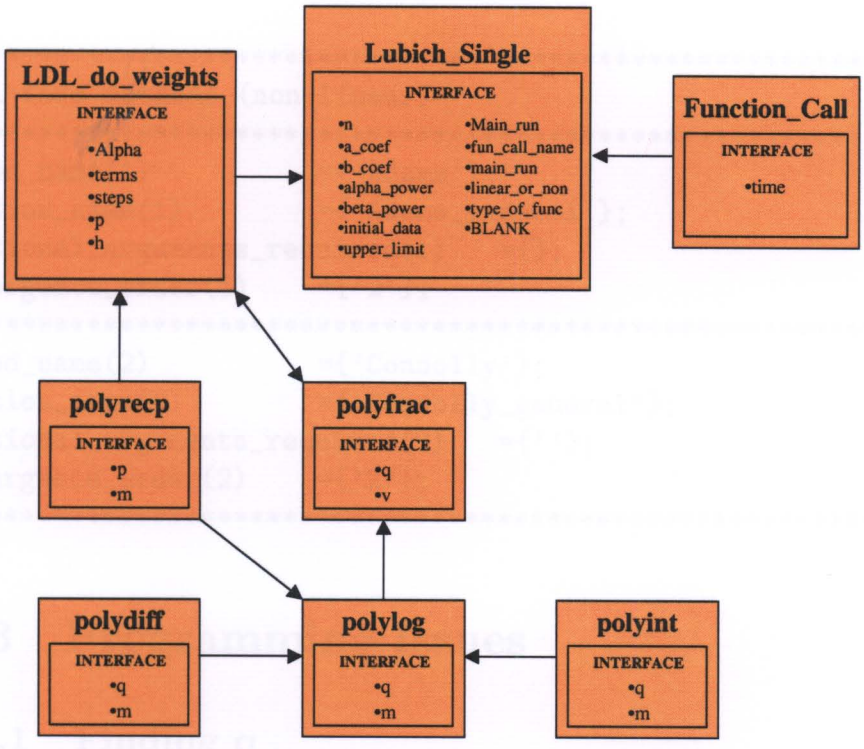

Figure 12.7: Lubich_Single model

12.2 Substituting Modules and Inserting New Methods

Most of the modules represented by rectangular boxes in figures 12.1 to 12.7 are stand alone objects. By this we mean that we can replace any module by just adhering to its interface.. The exceptions to this are 'FDE_GUI', 'RESULTS_GUI', 'MULTI_TERM_SOLVER', and 'DISTRIBUTED_SOLVER', as these actually write data directly to each other. These must be changed simultaneously.

Inserting New Methods

New methods can be inserted by adhering to either the single -term, multi-term or distributed interface. We then let the program know the name, convergence order and any additional arguments in the module 'SET_GUI_VARIABLES'. For example, for multi-term non-linear methods, we add an additional method using the following:

```
************************************************************
multi-term methods (non-linear)
************************************************************
method_name(1)          ={'Adams'};
function_name(1)        ={'Adams_general'};
additional_arguments_required{1}   ={};
convergence_order(1)    ={'2'};
************************************************************
method_name(2)          ={'Connolly'};
function_name(2)        ={'Connolly_general'};
additional_arguments_required{2}   ={''};
convergence_order(2)    ={'2'};
************************************************************
```

12.3 Programming Issues

12.3.1 Finding_q

This function finds the highest common divisor for a set of data. The first data entry is used as a starting value, checked for divisibility with the remaining data, if each entry is an integer (error tolerance .000001), the highest common divisor

is found. If the first check is not affirmative, the first data entry is divided by 2, and then the test repeated. This process continues for $n = 3, \ldots, 2000$ until a common divisor is found.

The limit of 2000 is enforced, to avoid the possibility of an endless loop.

```
%This function finds the highest common divisor for a set of data
function q=finding_q(coeffs)

is_integer=zeros(1,length(coeffs));
one_array=ones(1,length(coeffs));
k=1;
for k=1:2000,
    coff1_divisor=coeffs(1)/k;
    for i=1:length(coeffs),
        if mod(coeffs(i)/coff1_divisor,1)<0.000000001,
            is_integer(1,i)=1;
        end
    end

    if isequal(one_array,is_integer(1,:))==1,
        k_val=k
        q=coeffs(1)/k;
        return
    end
    k=k+1;
end
```

12.3.2 Efficient Function Calls

The time taken for Matlab to repeatedly call a function to perform a task is far greater if we use a 'for' loop. For linear functions, only a single function call is required. Instead of passing an individual time value, we pass an array of all the required time values. We then get an array of function values returned. This rapidly speeds up run-time.

```
function_value=zeros(1,steps-1);
switch type_of_function
    case 'symbolic',
        t=1/n:1/n:upper_limit;
        function_value(1,:)= eval(function_call_name);
    case {'name','hardcode'},
```

```
        t=1/n:1/n:upper_limit;
        function_value= feval(str2func(function_call_name),t)
                        /coefficients(length(coefficients));
end
```

12.3.3 Percentage Error Estimates

As the actual error for a particular solution is not known, we estimate the error. We do this by reducing the step size of a particular method by a factor of, for example, 8. We then use this approximate solution as an estimated exact solution. Using the following code we can then compute error bounds for the final solution.

```
for i=1:length(methods_used),
    if methods_used(i) >0,
        percentage_errorX2(i)=1/(((estimated_exact_n(i)
            /(2*method_n(i)))*2)^eval(char(convergence_order(i))));
        percentage_errorX3(i)=1/(((estimated_exact_n(i)
            /(4*method_n(i)))*2)^eval(char(convergence_order(i))));
        upper_errorX2(i)=(1+percentage_errorX2(i));
        upper_errorX3(i)=(1+percentage_errorX3(i));
        X2_upper(i)=X2(i)*upper_errorX2(i);
        X3_upper(i)=X3(i)*upper_errorX3(i);
        lower_errorX2(i)=(1-percentage_errorX2(i));
        lower_errorX3(i)=(1-percentage_errorX3(i));
        X2_lower(i)=X2(i)*lower_errorX2(i);
        X3_lower(i)=X3(i)*lower_errorX3(i);
    end
end
```

12.3.4 Gradients and Constants

In section 6.3 we introduced a method for comparing numerical methods relative efficiency. In this section we give code to estimate a particular methods 'path', using the error bounds from section 12.3.3 and basic trigonometry.

```
grad=zeros(3,length(methods_used));
const=zeros(3,length(methods_used));
for j=1:length(methods_used),
     grad(1,j)=(log(Y2(j))-log(Y3(j)))
```

```
        /(log(abs(X2_upper(j)))-log(abs(X3_upper(j))));
    grad(2,j)=(log(Y2(j))-log(Y3(j)))
        /(log(abs(X2(j)))-log(abs(X3(j))));
    grad(3,j)=(log(Y2(j))-log(Y3(j)))
        /(log(abs(X2_lower(j)))-log(abs(X3_lower(j))));
    const(1,j)=log(Y2(j))-grad(1,j)*log(abs(X2_upper(j)));
    const(2,j)=log(Y2(j))-grad(2,j)*log(abs(X2(j)));
    const(3,j)=log(Y2(j))-grad(3,j)*log(abs(X2_lower(j)));
end
```

12.3.5 Estimating Run Time

As the specific accuracy required is known, we can use the 'path' of the method, calculated as in section 12.3.4, to give estimates of run time, with error bounds depending upon the percentage error estimate from section 12.3.3. Then using the upper and lower estimates of run time, we show the likelihood of the method with the lowest estimated run time being the most numerically efficient .

```
log_Y2_best_method=grad(:,:).*critical_x_required+const(:,:);

best_method=zeros(1,3);
for i=1:3,
    [val,best_method(i)]=min(exp(log_Y2_best_method(i,:)));
end
if best_method(1)==best_method(2) & best_method(2)==best_method(3),
    %method with least time for upper, middle and lower
    the_best_method=best_method(3);
elseif best_method(2)==best_method(3),
    %method with least time for middle and lower
    the_best_method=best_method(1);
    split_decision='minor';
elseif best_method(2)~=best_method(3),
    the_best_method=best_method(3);
    split_decision='major';
end
```

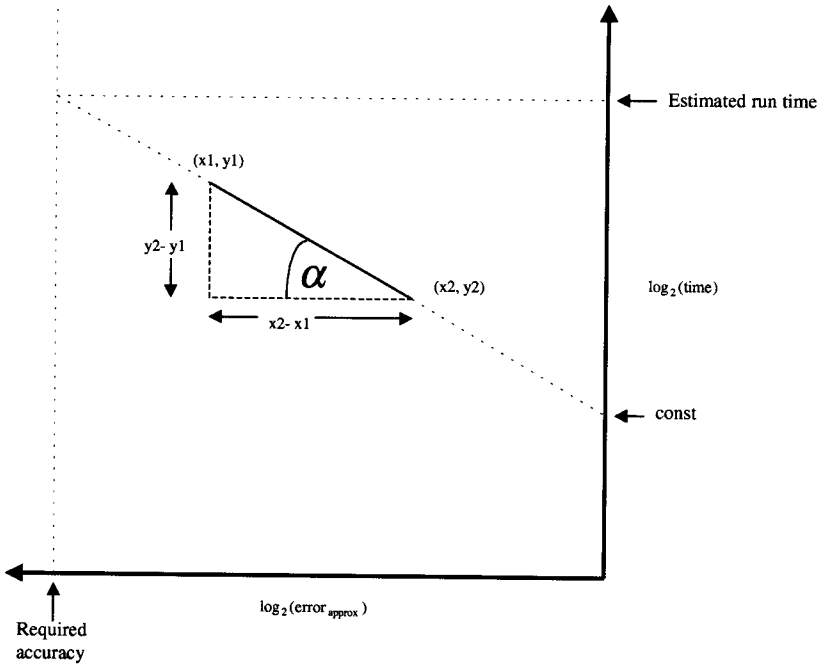

Figure 12.8: Calculating the gradient, constant and run time

Figure 12.8 illustrates how the estimated run time can be calculated, using the methods 'path'.

12.4 Conclusions

In this chapter we have shown an automated FDE solver capable of computing the most numerically efficient method for any given FDE or DODE. We have shown how the solver can be easily updated and illustrated some programming issues.

In the next chapter we introduce some current and future projects.

Chapter 13

Current and Future Projects

In this chapter we discuss future work in the area of fractional calculus and its applications. We discuss the concept of variable order differential equations and introduce a prototype method for their solution. Mathematical models using partial fractional differential equations and partial distributed order differential equations are discussed. Data fitting for FDE models is discussed and we illustrate how data fitting used in conjunction with the FDE solver, introduced in chapter 12, can enhance an engineer's ability to model FDEs.

13.1 Variable Order Equations

Lorenzo and Hartley [49] introduce the idea of a variable order Riemann-Liouville derivative $D_*^{q(t,\tau)}y(t)$. Three types of variability are explored, $q(t,\tau) = \alpha(t), q(t,\tau) = \alpha(\tau)$ and $q(t,\tau) = \alpha(t-\tau)$. In viscoelasticity the behaviour of certain materials can switch from elastic to viscoelastic or even to viscous, due to the effect of temperature fluctuations . Also, in viscoelasticity, the order of the derivative can be reliant on stress/strain ratios, and in electroviscous and electrorheological fluids the order of the derivative can be effected by electric fields.

Problems in these disciplines produce equations with no easy analytical solution. Thus the need for effective numerical methods for their solution is apparent. Up to now, no such algorithms exist.

We now introduce a prototype method for the solution of a variable order differential equation (VODE) in which the order of the method varies linearly with time t. The extension to non-linear variability is straightforward, so long as the order of the derivative $\alpha(t)$ is strictly increasing or decreasing.

13.1.1 Single-term VODE Solution Using Implicit Quadrature

We produce a method for solving the linear single-term (VODE),

$$D_*^{\alpha(t)}[y - y_0](t) = g(t) + \lambda y(t), \tag{13.1}$$

where $0 < \alpha(t) < m, m = \lceil \alpha \rceil$ subject to the initial conditions

$$y^{(k)}(0) = y_0^{(k)}, k = 0, \ldots, m-1.$$

Here D_* is the Riemann-Liouville fractional derivative. By subtracting the initial condition inside the derivative, D_* is identical to the Caputo fractional derivative D, thus avoiding problems regarding initial conditions.

We let α vary linearly with t from $a \to b$, where $a < b$,

thus $\alpha(j/n) = a + \frac{(b-a)j}{n}$.

Our method is based on the implicit quadrature scheme introduced in section 5.1.

The variable order fractional derivative can be written as

$$D_*^{\alpha(t)}y(t) = \frac{1}{\Gamma(-\alpha(t))}\int_0^t \frac{y(u)}{(t-u)^{\alpha(t)+1}}du$$

where as in chapter 5, the integral is Hadamard finite-part.

Discretizing on an equispaced grid $t_j = j/n$ for $j = 1, 2, \ldots, n$ and applying equation (13.1) gives,

$$\begin{aligned} g(t_j) + \lambda y(t_j) &= \frac{1}{\Gamma(-\alpha(t_j))}\int_0^{t_j} \frac{y(u) - y_0}{(t_j - u)^{\alpha(t_j)+1}}du \\ &= \frac{t_j^{-\alpha(t_j)}}{\Gamma(-\alpha(t_j))}\int_0^1 \frac{y(t_j - t_j\tau) - y_0}{\tau^{\alpha(t_j)+1}}d\tau. \end{aligned}$$

We replace the integral with a first degree compound quadrature formula,

$$\sum_{i=0}^{j} w_{ij}g(i/j) \approx \int_0^1 g(u)u^{-\alpha(t_j)-1}du,$$

where weights w_{ij} are,

$$\Phi(\alpha,n,j)w_{ij} = \begin{cases} -1 & \text{for } i=0, \\ 2i^{1-\alpha(\frac{j}{n})} - (i-1)^{1-\alpha(\frac{j}{n})} - (i+1)^{1-\alpha(\frac{j}{n})} & \text{for } i=1,2,\ldots,\text{j-1}, \\ (\alpha(\frac{j}{n})-1)i^{\alpha}(\frac{j}{n}) - (i-1)^{1-\alpha(\frac{j}{n})} + i^{1-\alpha(\frac{j}{n})} & \text{for } i=j, \end{cases} \tag{13.2}$$

where

$$\Phi(\alpha,n,j) = \alpha(j/n)\left(1-\alpha(j/n)\right) j^{-\alpha(j/n)}.$$

We represent the approximations $y(t_j), j=1,2,\ldots,n$ by y_j.

Thus the equation (13.1) can be solved using,

$$y_j = \frac{1}{w_{0j} - (j/n)^{\alpha(j/n)}\Gamma(-\alpha(j/n))\lambda} \left(\left(\frac{j}{n}\right)^{\alpha(j/n)} \Gamma\left(-\alpha(j/n)\right) g\left(t_j\right) - \sum_{i=1}^{j} w_{ij} y_{j-i} - \frac{1}{\alpha(j/n)} y_0 \right), \tag{13.3}$$

with weights (13.2).

13.2 Partial Fractional Differential Equations

In [26], Diethelm and Weilbeer discuss initial-boundary value problems for time-fractional diffusion-wave equations of the form,

$$D^{\alpha}_{t,*}y(x,t) + \phi(x,t)\frac{\partial^2}{\partial y^2}y(x,t) = f(x,t), \tag{13.4}$$

where $D^{\alpha}_{t,*}$ denotes the Caputo differential operator defined by

$$D^{\alpha}_{t,*}y(x,t) = \frac{1}{\Gamma(1-\alpha)}\int_0^t (t-\tau)^{-\alpha}\frac{\partial}{\partial\tau}y(x,\tau)d\tau.$$

The mathematics of these equations is not yet well understood. Only low order numerical methods have been produced [52] for their solution, which, because of the multiple dimension of partial differential equations, are computationally expensive. Higher order methods are needed to reduce the computational complexity of PFDEs.

13.3 Partial Distributed Order Differential Equations

In [8], Chechkin et al. introduce the partial distributed order differential equation (PDODE)

$$\int_0^1 d\beta\tau^{\beta-1}p(\beta)\frac{\partial^{\beta}f}{\partial t^{\beta}} = D\frac{\partial^2 f}{\partial x^2}, f(x,0) = \delta(x), \tag{13.5}$$

to model diffusion.

Where:

$\tau > 0$ represents some characteristic time of the problem.

D is the diffusion constant.

$p(\beta)$ is a dimensionless non-negative function.

β is a Caputo derivative.

Partial distributed order differential equations have only recently been introduced. No numerical methods for their solution are currently available.

13.4 Data Fitting

In chapter 12 we describe a computer program designed to find the quickest numerical method for a given equation (either single-term, multi-term or DODE).

When modelling a particular phenomenon an engineer however, may not be sure what the correct equation is. More specifically the orders of any fractional derivatives involved and the values of any constants may need to vary to accurately mimic the physically observed behaviour.

In section 10.12 of [56], Podlubny gives a description of FDE data fitting. Having an experimentally determined data set, Podlubny uses a method of least squares and then an optimization technique to determine a set of constants, which can be used in a FDE model.

For example, we fit a set of data points,

$$y_1, y_2, \ldots, y_n,$$

to satisfy the integral equation

$$y(t) = \sum_{k=0}^{m-1} a_k t^k - a_m D^{-\alpha} y(t), \qquad (0 < \alpha \leq m), \tag{13.6}$$

so as to determine the constants

$$a_k, (k = 0, \ldots, m). \tag{13.7}$$

Due to the physical interpretation of integer order derivatives, the problem can be reduced to the initial value problem,

$$D^{\alpha} z(t) + a_m z(t) = -a_m \sum_{k=0}^{m-1} a_k t^k,$$

$$z^{(k)}(0) = 0, \qquad (k = 0, \ldots, m-1),$$

for the auxiliary unknown function z(t), where

$$z(t) = y(t) - \sum_{k=0}^{m-1} a_k t^k. \tag{13.8}$$

For a fixed value of α and a combination of parameters (13.7), a solution $z(t)$ is determined.

Backward substitution using equation (13.8) is performed and the least squares criterion for the function $y(t)$ is evaluated. Optimization methods are then used to determine the optimum set of parameters for (13.7).

The set of parameters is then used to compute a solution to equation (13.6). This solution can then be analysed by the engineer, adjustments made to the values of α, k and the parameters (13.7), and the process restarted. The iteration procedure continues until an adequate model is obtained (see Podlubny [56] for further details).

Much of this process can be automated in a computer program. In section 6.4 we showed that the most numerically efficient method for a FDE does not alter when the parameter values (13.7) change. If the function $g(t)$ represented in (13.6) by t^k, or the value of α change however, the most numerically efficient method for a FDE does alter.

A data fitting program, used in tandem with the FDE solver introduced in this thesis, would therefore enhance an engineer's ability to use FDEs as a modelling tool.

13.5 Conclusions

In this chapter we have discussed variable order differential equations (VODEs), partial fractional differential equations and partial distributed order differential equations. A prototype method for the solution of VODEs is introduced.

The FDE solver introduced in chapter 12 can be extended to include each of these classes of equation. As new methods are produced for the solution of these equations they can be inserted into the FDE solver and the graphical technique introduced in section 6.3 used to determine the most numerically efficient method for any given equation.

Appendix A

Mathematics

In this appendix we describe some of the numerical methods for the solution of ordinary differential equations (ODEs) that have been used in the production of methods for the solution of fractional differential equations . Additional information on mathematical techniques can be found in [3], [9], [38], [41], [42] and [59].

A.1 Collocation [41] 194

In collocation we choose a function, a set of data points (collocation points), and demand that the function mimic the behaviour of the unknown function at these points. To solve an ordinary differential equation of the form

$$y' = f(x, y), \text{ where } y(a) = \eta,$$

we choose a polynomial P of degree s, a set of collocation points $\{x_n + c_i h, i = 1, 2, \ldots, s\}$, demanding that

$$P(x_n) = y_n,$$

$$P'(x_n + c_i h) = f(x_n + c_i h, P(x_n + c_i h)), i = 1, 2, \ldots, s.$$

We then take $y_{n+1} = P(x_n + h)$. The process is equivalent to an implicit s-stage Runge-Kutta method.

A.2 Numerical Solution of Ordinary Differential Equations

It is common practice to solve integer-order equations with a highest derivative greater than 1, using the following:

For the ODE

$$D^n y(t) + b_{n-1} D^{n-1} y(t) + \ldots + b_1 y(t) = g(t), y^{(i)}(0) = y_0^{(i)}, i = 0, \ldots, n-1, \quad \text{(A.1)}$$

we let

$$\begin{aligned} {}^1Y &= y, \\ &\vdots \\ {}^{i+1}Y &= D^i y, \quad i = 1, \ldots, n. \end{aligned} \quad \text{(A.2)}$$

Expressing A.2 in matrix form gives

$$\begin{pmatrix} D & 0 & \ldots & 0 \\ 0 & & & \\ \vdots & & \ddots & \vdots \\ & & & 0 \\ 0 & \ldots & 0 & D \end{pmatrix} \begin{pmatrix} {}^1Y \\ \vdots \\ {}^nY \end{pmatrix} = \begin{pmatrix} {}^2Y \\ \vdots \\ -\sum_{i=1}^n b_i \, {}^iY + g(t) \end{pmatrix}. \quad \text{(A.3)}$$

A solution y to equation (A.1) can be obtained by discretising the derivatives of the matrix on the left hand side of (A.3) and then solving the resulting system.

A.3 Mathematical Writing

There are many books available for advice on mathematical writing, one of the best is that of Higham [37]. This thesis was produced using the LaTeX computer language, good books on the structure of the language are that of Higham [36] and Kopka and Daly [40].

Appendix B

Glossary of Terms

J^{α} = Riemann-Liouville fractional integral

D_{*}^{α} = Riemann-Liouville fractional derivative

$\tilde{D}^{\alpha}$ = Grünwald-Letnikov fractional derivative

D^{α} = Caputo fractional derivative

h = the step size of the a single or multi-term method

t = time (secs)

y_j = the approximation of $y(jh)$ at integer multiples of the step length

$g(t)$ = the linear right hand side function

$f(t, y(t))$ = the non-linear right hand side function

λ = the coefficient multiplying the y term of the single-term FDE

m = the number of corrector iterations

α = the order of the fractional derivative

w_{ij} = the weights of the implicit quadrature method

b_{ij} = the weights of the predictor

a_{ij} = the weights of the corrector

p = convergence order of a method

$\tilde{y}$ = the analytical solution

$\hat{y}_j$ = the approximation to the homogeneous case of the approximate Mittag-Leffler method

$\tilde{y}_j$ = the approximation to the inhomogeneous case of the approximate Mittag-Leffler method

$\omega(k;h)$ = are the coefficients of a power series

T = a fixed time interval

$u(t_{j,i})$ = a collocation spline

V^i = the weight matrix of a collocation technique

$E_\alpha(t)$ = the Mittag-Leffler function of one variable

$E_{\alpha,\beta}(t)$ = the Mittag-Leffler function of two variables

$E(\beta,t)$ = the multi-variant Mittag-Leffler function

Γ = the Gamma function

$G(t)$ = the Green's function

RT_{est} = run time estimate

$M = \lceil \alpha \rceil - 1$

H = the step size of the distribution discretization

M1 = the error associated with the discretization of the distributed order

M2 = the error associated with the underlying FDE method

N = the number of nodes used in discretising the distribution integral

Acronyms

ODE - ordinary differential equation

FDE - fractional differential equation

DODE - distributed order differential equation

VODE - variable order differential equation

PFDE - partial fractional differential equation

PDODE - partial distributed order differential equation

GUI - graphical user interface

EOC - experimentally determined order of convergence

EOC_N - experimentally determined order of convergence of the M1 error

DDM - distributed Diethelm method

DAM - distributed Adams method

AEE - asymptotic error expansion

ETR - extended trapezoidal rule

GAM - generalized Adams method

DTM - distributed trapezoidal method

PCEC - predict correct evaluate correct

Bibliography

[1] R. L. Bagley and P. J. Torvik. On the appearance of the fractional derivative in the behaviour of real materials. *J. Appl. Mech.*, 51:294–298, 1984.

[2] L. Blank. Numerical treatment of differential equations of fractional order. Technical report, Manchester Centre for Computational Mathematics: Numerical Analysis Report 287, 1996.

[3] L. Brugnano and D. Trigiante. *Solving Differential Problems by Multistep Initial and Boundary Value Methods.* Gordon and Breach Science Publishers, 1998.

[4] H. Brunner, Q. Hu, and Q. Lin. Geometric meshes in collocation methods for Volterra integral equations with proportional delays. *IMA Journal of Numerical Analysis*, 21:783–798, 2001.

[5] M. Caputo. Linear models of dissipation whose Q is almost frequency independent-II. *Geophys. J. R. astr. Soc.*, 13:529–539, 1967.

[6] M. Caputo. Distributed order differential equations modelling dielectric induction and diffusion. *Fractional Calculus and Applied Analysis*, 4:421–442, 2001.

[7] A. V. Chechkin, R. Gorenflo, and I. M. Sokolov. Retarding subdiffusion and accelerating superdiffusion governed by distributed-order fractional diffusion equations. *Physical Review*, 66, 2002.

[8] A. V. Chechkin, R. Gorenflo, I. M. Sokolov, and V. Y. Gonchar. Distributed order time fractional diffusion equation. *Fractional Calculus and Applied Analysis*, 6:259–279, 2003.

[9] R. V. Churchill. *Operational Mathematics.* McGraw-Hill, 1958.

[10] L. Debnath. Fractional integral and fractional differential equations in fluid mechanics. *Fractional Calculus and Applied Analysis*, 6:119–137, 2003.

[11] K. Diethelm. An algorithm for the numerical solution of differential equations of fractional order. *Electronic Transactions on Numerical Analysis*, 5:1–6, 1997.

[12] K. Diethelm. Generalized compound quadrature formulae for finite-part integrals. *IMA Journal of Numerical Analysis*, 17:479–493, 1997.

[13] K. Diethelm, J. M. Ford, N. J. Ford, and M. Weilbeer. Pitfalls in fast numerical solvers for fractional differential equations. *submitted to Journal of computational and applied mathematics*, to appear.

[14] K. Diethelm and N. J. Ford. Numerical solution methods for distributed order differential equations. *Fractional Calculus and Applied Analysis*, 4:531–542, 2001.

[15] K. Diethelm and N. J. Ford. Numerical solution of linear and non-linear fractional differential equations involving fractional derivatives of several orders. *Manchester Centre for Computational Mathematics Numerical Analysis Report 379*, 2001.

[16] K. Diethelm and N. J. Ford. Analysis of fractional differential equations. *Journal of Mathematical Analysis and Applications*, 265:229–248, 2002.

[17] K. Diethelm and N. J. Ford. Numerical solution of the Bagley Torvik equation. *BIT*, 42 no. 3:490–507, 2002.

[18] K. Diethelm and N. J. Ford. Multi-order fractional differential equations and their numerical solution. *Applied Mathematics and Computation*, 154:621–640, 2004.

[19] K. Diethelm, N. J. Ford, and A. D. Freed. A predictor-corrector approach for the numerical solution of fractional differential equations. *Nonlinear Dynamics*, 29:3–22, 2002.

[20] K. Diethelm, N. J. Ford, and A. D. Freed. Detailed error analysis for a fractional Adams method. *Numerical Algorithms*, 36:31–52, 2004.

[21] K. Diethelm, N. J. Ford, A. D. Freed, and Y. Luchko. Algorithms for the fractional calculus: A selection of numerical methods. *Computer Methods in Applied Mechanics and Engineering*, 196:743–773, 2005.

[22] K. Diethelm and A. Freed. Tensor fields for use in fractional-order viscoelasticity. *Proceedings of the 1st IFAC Workshop on Fractional Differentiation and its Applications. ENSEIRB, Bordeaux*, pages 86–91, 2004.

[23] K. Diethelm and A. D. Freed. The FracPECE subroutine for the numerical solution of differential equations of fractional order. *In S. Heinzel, T. Plesser (eds.): Forschung und wissenschaftliches Rechnen: Beiträge zum Heinz-Billing-Preis*, 1998.

[24] K. Diethelm and Y. Luchko. Numerical solution of linear multi-term initial value problems of fractional order. *Fractional Calculus and Applied Analysis*, to appear.

[25] K. Diethelm and G. Walz. Numerical solution of fractional order differential equations by extrapolation. *Numerical Algorithms*, 16:231–253, 1997.

[26] K. Diethelm and M. Weilbeer. Initial-boundary value problems for time-fractional diffusion-wave equations and their numerical solution. *Proceedings of the 1st IFAC Workshop on Fractional Differentiation and its Applications. ENSEIRB, Bordeaux*, pages 551–557, 2004.

[27] N. J. Ford and J. A. Connolly. Comparison of numerical methods for fractional differential equations. *Accepted for publication, Communications on pure and applied analysis*, 2005.

[28] N. J. Ford, J. T. Edwards, and A. C. Simpson. The numerical solution of linear multi-term fractional differential equations: Systems of equations. *Journal of Computational and Applied Mathematics*, 148:401–418, 2001.

[29] N. J. Ford and A. C. Simpson. The approximate solution of fractional differential equations of order greater than 1. Technical report, Manchester Centre for Computational Mathematics: Numerical Analysis Reports 386, 2001.

[30] N. J. Ford and A. C. Simpson. Numerical and analytical treatment of differential equations of fractional order. Technical report, Manchester Centre for Computational Mathematics: Numerical Analysis Reports 387, 2001.

[31] N. J. Ford and A. C. Simpson. The numerical solution of fractional differential equations: speed versus accuracy. *Numerical Algorithms*, 26:333–346, 2001.

[32] A. D. Freed, K. Diethelm, and Y. Luchko. Fractional-order viscoelastic (FOV): Constitutive development using the fractional calculus: First annual report. Technical report, John H. Glenn Research Center E-13607, 2002.

[33] R. Gorenflo. Fracional calculus: Some numerical methods. *Fractals and Fractional Calculus in Continuum Mechanics, Springer Verlag, New-York and Wien*, pages 277–290, 1997.

[34] R. Gorenflo and F. Mainardi. Feller fractional diffusion and levy stable motions. In *Conference on Levy Processes: Theory and Applications*. University of Aarhus, Denmark, 1999.

[35] R. Gorenflo and R. Rutman. On ultraslow and on intermediate processes, transform methods and special functions. *SCT Publishers, Singapore*, pages 61–81, 1994.

[36] D. F. Griffiths and D. J. Higham. *learning LATEX*. The Society for Industrial and Applied Mathematics, 1997.

[37] N. J. Higham. *Handbook of Writing for the Mathematical Sciences*. Society for Industrial and Applied Mathematics, 1998.

[38] A. Iserles. *A First Course in the Numerical Analysis of Differential Equations*. The Press Syndicate of the University of Cambridge, 1996.

[39] M. Kleinz and T. Osler. A child's garden of fractional derivatives. *The College Mathematics Journal*, 2000.

[40] H. Kopka and P. W. Daly. *A Guide To LATEX*. Addison-Wesley, 1999.

[41] J. D. Lambert. *Numerical Methods for Ordinary Differential Equations*. John Wiley and Sons Ltd., 1991.

[42] P. Linz. *Analytical and Numerical Methods for Volterra Equations.* Society for Industrial and Applied Mathematics, 1985.

[43] C. F. Lorenzo and T. T. Hartley. Initialization, conceptualization, and application in the generalized fractional calculus. Technical report, NASA/TM-1998-208415, 1998.

[44] C. F. Lorenzo and T. T. Hartley. Insights into the fractional order initial value problem via semi-infinite systems. Technical report, NASA/TM-1998-208407, 1998.

[45] C. F. Lorenzo and T. T. Hartley. A solution to the fundamental linear fractional order differential equation. Technical report, NASA/TM-1998-208693, 1998.

[46] C. F. Lorenzo and T. T. Hartley. Fractional system identification: An approach using continuous order-distributions. Technical report, NASA/TM-1999-209640, 1999.

[47] C. F. Lorenzo and T. T. Hartley. The vector linear fractional initialization problem. Technical report, NASA/TM-1999-208919, 1999.

[48] C. F. Lorenzo and T. T. Hartley. R-function relationships for application in the fractional calculus. Technical report, NASA/TM-2000-210361, 2000.

[49] C. F. Lorenzo and T. T. Hartley. Variable order and distributed order fractional operators. *Nonlinear Dynamics*, 29:57–98, 2002.

[50] C. Lubich. Convolution quadrature and discretized operational calculus II. *Numerical Mathematics*, 52:413–425, 1988.

[51] Y. Luchko and R. Gorenflo. The initial value problem for some fractional differential equations with the caputo derivatives. Technical report, University of Berlin, A98-08, 1998.

[52] V. E. Lynch, B. A. Carreras, D. del Castillo-Negrete, K. M. Ferreira-Mejias, and H. R. Hicks. Numerical methods for the solution of partial differential equations of fractional order. *Journal of Computational Physics*, 192:406–421, 2003.

[53] F. Mainardi. Fractional calculus: Some basic problems in continuum and statistical mechanics. *Fractals and Fractional Calculus in Continuum Mechanics, CISM Courses and Lectures, no. 378, Springer, Wien*, pages 291–348, 1997.

[54] K. S. Miller and B. Ross. *An Introduction to the Fractional Calculus and Fractional Differential Equations.* John Wiley and Sons, 1993.

[55] I. Podlubny. Numerical solution of ordinary fractional differential equations by the fractional difference method. In *Proceedings of the Second International Conference on Difference Equations, (eds.:S.Elaydi, I. Gyori, G. Ladas)*, pages 507–516. Gordon and Breach Science Publishers, 1997.

[56] I. Podlubny. *Fractional Differential Equations.* Academic Press, San Diego, 1999.

[57] I. Podlubny. Matrix approach to discrete fractional calculus. *Fractional Calculus and Applied Analysis*, 3:359–386, 2000.

[58] I. Podlubny. Geometric and physical interpretation of fractional integration and fractional differentiation. *Fractional Calculus and Applied Analysis*, 5:367–386, 2002.

[59] A. Quarteroni, R. Sacco, and F. Saleri. *Numerical Mathematics.* Springer, 2000.

[60] S. G. Samko, A. A. Kilbas, and O. I. Marichev. *Fractional Integrals and Derivatives: Theory and applications.* Gordon and Breach Science Publishers, 1993.

[61] A. Schmidt and L. Gaul. FE implementation of viscoelastic constitutive stress-strain relations involving fractional time derivatives. Technical report, University of Stuttgart, Germany, 2000.

[62] C. Simpson. *Numerical Methods for the Solution of Fractional Differential Equations.* PhD thesis, University of Liverpool, 2001.

[63] L. Yuan and O. P. Agrawal. A numerical scheme for dynamic systems containing fractional derivatives. *Proceedings of DETC98 1998 ASME Design Engineering Technical Conferences, Atlanta, Georgia*, 1998.